MASTER EN SCIENCES INFORMATIQUES

Évaluation des performances d'un système d'exploitation : une approche par simulation

Parizel, René

Award date:
1974

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX A NAMUR

Institut d'Informatique

Année académique 1973-1974

EVALUATION DES PERFORMANCES D'UN SYSTEME D'EXPLOITATION

- UNE APPROCHE PAR SIMULATION -

René PARIZEL

Jury de mémoire :
Monsieur J. RAMAEKERS

Mémoire présenté en vue de l'obtention du grade de Licencié et Maître en Informatique.

Nous tenons à remercier Monsieur F. BODART, Directeur de l'Institut d'Informatique, ainsi que les membres de ce même Institut, pour la formation que nous avons acquise au cours de ces années de Licence et Maîtrise en Informatique.

Nous remercions vivement tous ceux qui par une aide directe ou une collaboration plus effacée, mais toujours bienvenue et efficace, ont rendu possible la réalisation de ce travail.

Nous exprimons notre entière gratitude envers Monsieur J. RAMAEKERS, qui a bien voulu diriger et orienter cette étude.

Nous assurons de notre profonde reconnaissance les membres de la Division "Etudes et Développements Spéciaux" de la C.H.B. - Paris ; en particulier, Monsieur P.H. RIVET de SABBATIER, responsable des Etudes de Performances, et Monsieur J.P. CORRE, qui ont grandement facilité l'élaboration du contenu de ce mémoire.

Un grand merci à Mademoiselle Jeanine de Fays, pour son extrême amabilité et le soin minutieux apporté à la dactylographie de ce travail.

Namur, le 2 mai 1974.

T A B L E D E S M A T I E R E S

TABLE DES MATIERES

I N T R O D U C T I O N

I N T R O D U C T I O N

Lorsque l'on aborde le domaine de l'informatique, chacun s'accorde à reconnaître, en toute évidence d'ailleurs, que cette nouvelle application des sciences n'a pas encore atteint, loin s'en faut, son âge mûr. Dieu sait cependant si la gamme des moyens qu'elle offre actuellement et la taille de certains problèmes à la résolution desquels elle a largement contribué, sont impressionnantes.

Néanmoins, et en dépit de ses magnifiques réalisations, le traitement automatique de l'information est confronté depuis quelques années à un nouveau type de problème, inhérent celui-là aux techniques mêmes qu'il utilise : celui des performances des configurations en place.

Le contenu des chapîtres qui vont suivre se propose d'aborder cette question de façon très modeste, étant donné que :

1) il est hors de propos d'apporter ici une solution complète à un problème aussi vaste.
2) l'examen de l'évaluation d'une configuration sera réduit à celui des moyens disponibles pour estimer les performances d'un operating system.
3) parmi ces moyens, alors que certains seront cités simplement à titre informatif, une approche par simulation sera développée un peu plus en détail, et ce dans une optique particulière précisée dans la suite.

Une lecture globale des articles écrits sur le sujet qui nous préoccupe débouche rapidement sur la conclusion suivante : l'auteur insiste très souvent sur la nécessité de procéder à une étude des performances d'un système d'exploitation ; ceci pour des motifs apparemment différents mais à partir desquels se dégage facilement

un but commun. A ce propos, il est intéressant d'analyser brièvement la position, parmi une multitude d'auteurs, de trois d'entre eux :

Goldberg- Huesmann [1]

"As computer systems have grown in size and computer implementations have grown in sophistication, it has became increased more important to develop means for evaluating different hardware and software configurations both prior to and after installation. One approach to such evaluation has been through digital computer simulation of proposed computer systems. ..."

Lucas [2] :

"The evaluation of computer performance is of vital importance in the selection of computer systems, the design of application and equipment, and the analysis of existing systems. Evaluation is usually undertaken for a specific purpose, and the techniques employed must be considered in light of the objectives of the evaluator. The goals of the evaluation may be to maximize the throughput of a system, process a given workload for a minimum cost, and any number of other objective functions. These goals furnish the overall environment for evaluation and determine what level of effort can be devoted to the measurement of performance. ..."

Il est certain qu'au cours de l'évolution des ordinateurs, celle des systèmes d'exploitation n'a pas été la moindre : en effet, entre la non-existence d'un O.S. et sa complexité qu'on lui connaît actuellement, le pas franchi est assez remarquable. Dans l'esprit du constructeur, le souci de libérer l'utilisateur de contraintes orientées "machine" et celui de lui fournir une variété de plus en plus grande de fonctions diverses et plus ou moins complexes est resté prédominant. Il suffit pour s'en convaincre de consulter l'ouvrage "OPERATING SYSTEMS SURVEY" (Appendix A) [16] qui à cet

égard donne un aperçu assez édifiant des rôles d'un système d'exploitation. Il résulte de ceci qu'au stade actuel, la conception, le développement et la mise au point d'un tel produit exigent en hommes et en temps des investissements considérables rendus nécessaires par l'étendue du travail à fournir. L'accomplissement de celui-ci étant basé sur une structure modulaire, l'aspect de contrôle des différentes entités séparées d'une part, de leur déroulement simultané d'autre part, prend une importance vitale au niveau de l'évaluation des performances.

H.C. Lucas [2] affirme de son côté que celle-ci est à mettre en étroite relation avec les desseins de celui qui la réalise. En d'autres termes, aussi nécessaire que soit cette opération, elle n'est jamais entreprise qu'en fonction d'un certain nombre d'objectifs que la concepteur ou l'utilisateur voudrait voir atteints par l'O.S.

Pour en revenir à présent au but commun dont la notion est mentionnée ci-dessus, on peut retenir de ces deux textes une idée fondamentale qui s'exprimerait de la façon suivante : quel que soit le motif invoqué pour procéder à l'évaluation des performances d'un système, à savoir apporter une certaine clarté au sein de la complexité des modules et de leurs relations, ou contrôler l'efficacité de ce système lorsqu'il doit mener à bien une tâche spécifique, il s'agit en fait de s'assurer la maîtrise du produit élaboré, de façon que l'élément humain puisse toujours, avec le maximum de conscience et de responsabilité quant aux conséquences futures, prendre une option destinée à rendre l'outil plus adapté au travail pour lequel il a été conçu, mis au point et amélioré.

*

* *

C H A P I T R E I :

DOMAINE CONSIDERE ET MOYENS CHOISIS

1.1. Concept de performance

1.2. Evaluation : quand et comment ?

1.3. La technique de simulation

1.4. Restrictions et objectif fondamental de l'étude

1.5. Schéma global

1.1. CONCEPT DE PERFORMANCE (1)

1.1.1. Signification

Il est toujours intéressant, avant d'entrer dans une étude particulière, de cerner le mieux possible les différentes notions qu'elle met en jeu. C'est pourquoi l'ensemble de ce sous-chapitre (1.1.) aura pour but de donner au concept de performance, une signification et une portée qui s'inscrivent dans une démarche et un cadre auxquels on veut attribuer une dimension réaliste.

Les performances d'un système peuvent se définir comme étant l'efficacité constatée de celui-ci lorsqu'on lui confie l'exécution d'un type de travail pour lequel il a été conçu. Une conséquence directe de cette approche fait apparaître l'aspect relatif (souligné également par Calingaert [4]) de la notion envisagée. Il n'est guère concevable en effet d'exiger d'un système des performances dites "absolues", qui ne sauraient tenir compte du caractère propre des applications soumises au traitement automatique. A titre d'exemple, il est illusoire d'attendre d'un système unique :

- un rendement optimal en temps réel,
- combiné avec un throughput maximum en batch-processing, tels qu'on serait en droit de les demander si l'on envisageait d'effectuer les deux traitements sur des systèmes séparés. Il est clair que la nature même des opérations que l'on veut réaliser va engendrer un type de performances correspondant, et qu'une amélioration, voire une optimisation de celles-ci ne peut être entreprise ni menée à bien sans prendre en considération les objectifs en vue desquels le système a été construit.

---

(1) Ce sous-chapitre s'inspire de Graham [3]

1.1.2. Mesures de performances

Pour apprécier l'efficacité d'un système, il sera nécessaire de disposer d'un ensemble de valeurs traduisant les performances obtenues. Un problème se pose de suite : quelles sont les variables que l'on estime dignes d'intérêt, c'est-à-dire celles qui en somme sont les plus représentatives du service que le système a rendu ?
A ce sujet on peut distinguer deux raisonnements :

a) le premier a tendance à considérer le computer system comme une "boîte noire" à l'intérieur de laquelle jouent des mécanismes que l'on ne considère pas. L'intérêt se porte surtout au niveau de ce que l'on donne comme charge de travail, et des conditions dans lesquelles on reçoit les résultats (fig. 1.1). C'est ici que des variables comme le throughput ou le turnaround time par exemple trouvent leur raison d'être.

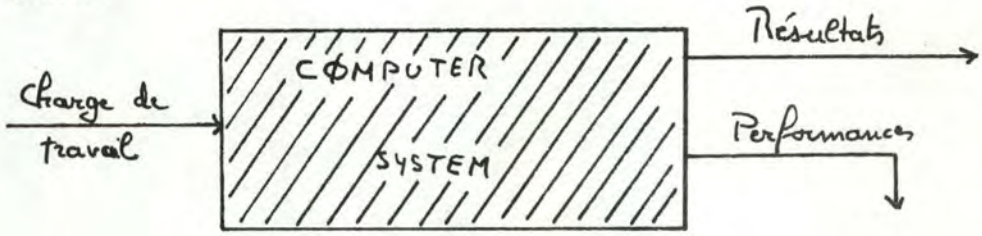


fig. 1.1.

Il va de soi que pour pouvoir effectuer une analyse plus ou moins approfondie des performances, un tel niveau de détail est largement insuffisant, et que la négligence systématique des mécanismes internes constitue un sérieux handicap à la possibilité de tirer des conclusions significatives.

b) le second s'attachera bien entendu à pénétrer à l'intérieur du computer system et à observer ce qui se passe réellement. La charge de travail est en fait interprétée comme une suite d'appels à

pouvoir disposer de certaines ressources (organes de traitement au sens large et organes de mémorisation - fig. 1.2) en quantité et temps voulus.
Ce sont les variables dérivant des occupations de ressources qui vont être cette fois prises en compte.
De façon plus précise, et à titre exemplatif, on considérera :

- à un instant t, le pourcentage de ressources concernées par le traitement.
- la durée d'occupation d'une ressource.
- le nombre d'appels à une ressource.
- le temps de réponse minimum, moyen et maximum d'une ressource.

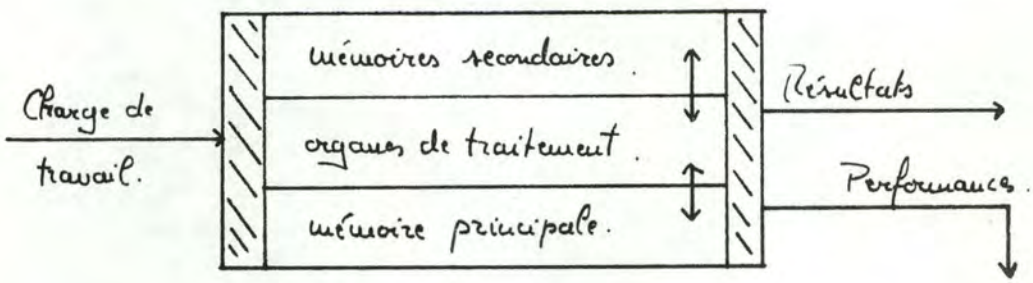


fig. 1.2.

Un examen rapide de ces deux raisonnements révèle ceci : bien qu'elles traduisent toutes une certaine efficacité, les variables citées en a) et en b) apparaissent comme naturelles respectivement à l'utilisateur et au concepteur ou gestionnaire du système. Le but final étant de satisfaire le mieux possible le premier, grâce à un ensemble de techniques que devra posséder le second, un lien entre les deux types de variables apparaît comme souhaitable. L'établissement de celui-ci de façon formalisée est assez difficile, et là ne se situe pas par ailleurs l'objectif primordial de cette étude.

Il reste néanmoins que l'évaluation des performances accordera la priorité à l'obtention de variables du

second type (dites de base) ; et ceci dans le dessein suivant : la connaissance d'une part de ce qui se passe au niveau des ressources prises individuellement et d'autre part des règles d'allocation de celles-ci aux différents processus constituera une base très valable en vue :

1) de déterminer les points faibles du système,
2) de procéder ensuite aux changements jugés opportuns à l'intérieur de celui-ci,
3) pour obtenir de meilleures performances en termes de variables de base, et par voie de conséquence, en termes de variables plus significatives pour l'utilisateur.

1.1.3. Limites des performances

Il est possible de déduire globalement cet aspect des choses de la notion présentée en 1.1.1., à savoir le principe des performances non absolues. Les contraintes qui imposent à celles-ci une certaine relativité peuvent entrer dans une des deux catégories suivantes :

a) Limites inhérentes aux principes de conception (soit de la configuration, soit du système et de ses fonctions).
Sans entrer dans le détail d'une liste qui pourrait être assez longue, on peut dégager des lignes de force donnant un aperçu général des éléments qui introduisent un "plafond" aux performances :

- l'utilisation de dispositifs physiques a pour conséquence immédiate l'apparition de la limite physique correspondante. Ceci intervient par exemple en termes de temps (pour la vitesse d'un signal, la réponse d'une ligne, ...) ou de place (capacité de mémoire principale, ...).

- un conflit d'utilisation d'une méthode ou d'une autre pour réaliser une fonction déterminée aboutira souvent à une solution de compromis espace mémoire - temps de traitement.
- la complexité d'un système dont le design promet monts et merveilles interdira pratiquement de manière automatique sa réalisation et son implémentation.

b) Limites économiques. Alors que les contraintes du premier type sont théoriquement impossibles à éliminer, celles-ci présentent la caractéristique de pouvoir être surmontées, mais à un prix tel que le rapport de coût - efficacité serait très défavorable.

A titre d'exemple on peut citer :

- l'utilisation d'un hardware à hautes performances, mais incompatible avec l'enveloppe budgétaire dont l'utilisateur peut disposer.
- la recherche intensive des algorithmes optimaux : en dépit du fait qu'ils assurent l'optimalité dans le domaine pour lequel ils sont conçus, il faut garder à l'esprit le fait que le temps consacré à trouver ces algorithmes, à les implémenter ensuite, sans parler encore de celui nécessaire à l'exploitation pour les exécuter, ne se justifie peut-être pas, étant donné de nouveau, les moyens financiers disponibles pour arriver à une solution acceptable.

- - - - -

1.2. EVALUATION : QUAND & COMMENT ?

1.2.1. Quand procéder à l'évaluation ?

Le problème de l'estimation des performances se pose généralement dans des conditions différentes selon les besoins que l'utilisateur ressent ou les buts que le constructeur veut atteindre. Gotlieb [5] et Lucas [2] donnent à ce sujet quelques précisions utiles .

De façon globale, on peut distinguer plusieurs objectifs à la réalisation desquels l'évaluation des performances contribue de façon substantielle :

a) Sélection d'une configuration et d'un système :
Ce problème se pose en termes d'une correspondance à établir le plus justement possible entre les nécessités du traitement automatique de l'information et la gamme des produits disponibles sur le marché.

b) Conception d'un nouveau système. Les spécifications du design du nouveau produit sont une chose, la réalité que celui-ci représentera lorsqu'il sera implémenté aura peut-être un autre visage. La prévision de ses performances et de la possibilité réelle de sa construction doit contribuer à mieux cerner la nature des conséquences éventuelles de telle ou telle décision.

c) Reconfiguration ou adaptation d'un système existant.
A partir des données caractérisant l'efficacité du système en place, on tire des conclusions quant aux modifications possibles du hardware, ou à l'amélioration de certains modules du software. Le but étant toujours de mieux utiliser l'outil, et d'obtenir un rapport coût - efficacité raisonnable.

1.2.2. Techniques d'évalutation

Selon les moyens dont on dispose, le degré de difficulté que l'on veut aborder et les résultats espérés, il est possible d'opérer un choix parmi les nombreuses techniques existantes. Lucas [2] en reprend quelques-unes, et en dégage les lignes de force sans pour autant entrer dans le détail d'une réalisation.

Les principales d'entre elles sont les suivantes :

a) comparaison des temps d'accès mémoire, des temps nécessaires pour une addition : l'application de cette technique est relativement simple. Un des inconvénients majeurs réside dans une totale ignorance du software.

b) les "instructions mixes" : on s'attache à composer un "mix" d'instructions relatif à un problème d'une nature particulière (scientifique - de gestion).
De nouveau, ce type d'évaluation présente le désavantage, parmi d'autres, d'ignorer l'existence d'un O.S.

c) les modèles analytiques. Ils sont constitués par une représentation mathématique du système.

d) les "benchmarks". Il s'agit là de programmes existants rédigés dans un langage donné et exécutés sur la configuration sujette à l'évaluation.

e) les programmes synthétiques. Bien qu'ils soient codés et exécutés, ils présentent une différence essentielle avec les "benchmarks" en ce sens qu'ils ne constituent pas des entités exploitées de manière systématique.

f) le monitoring (hardware et software).

g) la simulation. Une approche de cette technique sera développée dans la suite.

Il est à noter que Calingaert [4] aborde ce problème des techniques d'évaluation d'une façon légèrement différente et plus globale.

Etant donné que l'objet de ce travail se situe au niveau d'un développement d'une évaluation de performances dans une optique de simulation, la liste ci-dessus est donnée à titre exemplatif. Les composants de celle-ci (mis à part le dernier) ne seront donc pas repris plus en détail par après.

1.2.3. Synthèse

Au regard de ce qui précède, il est possible de tirer une conclusion quant aux principales directions que peut prendre une évaluation de performances. En effet, les différentes techniques existantes s'attachent à constater l'efficacité d'un système, ou à tenter d'appréhender le comportement de ce système sur base de l'introduction de diverses hypothèses relatives à la structure et au fonctionnement de celui-ci.

En d'autres termes, on peut parler respectivement de "technique de mesure" ou de "technique de prévision" des performances. A titre d'illustration, la méthode de "benchmark" relève essentiellement du domaine de la mesure alors que la formalisation d'un système par sa mise en équations a pour but de prévoir le comportement du produit. Il serait cependant erroné de croire à une séparation complète entre ces deux tendances : bien qu'elles nécessitent une démarche différente, leur complémentarité apparaît comme très réaliste et digne d'être utilisée avec attention. Un exemple typique peut être cité à propos d'un modèle de simulation (Lucas [2], Noetzel [6]). Il existe deux façons de fournir au

modèle l'input dont il aura besoin pour caractériser les appels à l'utilisation de ressources et la génération de processus :

a) la première consiste à introduire les éléments sous forme de valeurs recherchées dans des distributions théoriques,

b) tandis que la seconde se basera sur un éventail de résultats obtenus par une technique de mesure appropriée (hardware monitoring - software monitoring).

Selon que l'on applique l'une ou l'autre de ces démarches, le modèle de simulation utilisant ces données sera de manière naturelle orienté respectivement vers la prévision des performances d'un système pour lequel les hypothèses de base seraient vérifiées, ou vers le dégagement du comportement global d'un système pour lequel on dispose déjà de résultats empiriques. C'est plutôt vers cet aspect des choses que la suite de ce travail sera orientée.

- - - - -

1.3. LA TECHNIQUE DE SIMULATION

1.3.1. Le cadre : évaluation d'un O.S. existant

Dans l'optique choisie, le domaine auquel s'applique l'évaluation des performances est celui d'un système d'exploitation réel en cours de réalisation ; toutefois la technique introduite pourra être utilisée ultérieurement lorsque le système existera en entier, pour son suivi et sa maintenance.

Comme signalé ci-dessus, cette technique est une forme de simulation, dont les caractéristiques seront développées dans la suite.

1.3.2. Buts de la simulation

"Evaluation de performances" est considéré dans le cadre de ce travail comme étant la possibilité de dégager des normes de comportement d'un operating system grâce à l'emploi d'un modèle de simulation. Ces normes de comportement caractérisent le fonctionnement global du système envisagé par l'intermédiaire de variables dont la gamme des valeurs représente l'efficacité de ce système. Un exemple - type serait la longueur d'une file d'attente occupée par un ou plusieurs processus en attente d'une ressource (CPU, canal, ...).

Il faut signaler néanmoins que là ne réside pas le seul avantage à retirer d'une optique de simulation. En effet, en raison de la possibilité de description que possède la technique, un point fondamental se situe au niveau de la compréhension globale du système évalué (Mac Dougall [9] ; Rehmann - Gangwere [7] ; Bell, Boehm, Watson [8]). La complexité grandissante de produits tels que les operating systems devient à ce point élevée qu'il n'est plus possible à l'homme de percevoir de façon détaillée les innombrables fonctions remplies

par le système et les relations qui existent entre elles. Il est donc indispensable, pour arriver à construire un modèle d'estimation fidèle et cohérent, de pouvoir dégager une vue synthétique du produit à évaluer. Une fois cette étape franchie, les résultats statistiques que livrera le modèle permettront d'approfondir et de détailler la compréhension des différentes fonctions du système. Ce qui est finalement la condition nécessaire pour procéder à une analyse des points faibles du produit et proposer des améliorations en rapport avec les objectifs choisis.

Il est clair que ces deux aspects d'une optique simulation ne sont pas indépendants l'un de l'autre ; en effet plus précise sera la vision des principes directeurs du système, plus facile et plus efficace sera leur introduction dans le modèle d'évaluation.

1.3.3. Concepts liés à la simulation

a) Simulation. Dans le contexte qui nous occupe, la simulation est envisagée comme l'utilisation d'un ordinateur à des fins de représentation d'un autre système. Déjà un problème se pose : quel est le niveau de détail qu'il est nécessaire d'affecter à cette représentation ? Comme souvent dans le domaine informatique, c'est une solution de compromis qui sera le plus vraisemblablement retenue (d'autant plus que la souplesse de la technique le permet assez facilement). En effet :

- si le niveau de détail est trop grand : outre les coûts de réalisation et de mise au point de simulateur, les coûts d'exploitation de celui-ci seront certainement non négligeables.
- si par contre on reste à un niveau trop général, les résultats obtenus peuvent induire en erreur

et donner lieu à une analyse incorrecte.
En effet, certains détails importants, s'ils sont pris en compte ensemble et non séparément, risquent, par l'existence même de l'agrégat, de perdre leur impact propre.

Cette technique fait donc intervenir deux entités bien distinctes : l'objet simulant et l'objet simulé. Le premier cité est concrétisé par un modèle établi dans un langage spécifique (GPSS - SIMSCRIPT ...) ou non (FORTRAN, ...) tandis que le second est constitué par le système sujet à l'évaluation.

b) Modèle de simulation. Bien qu'il ne soit pas question de développer longuement la notion de modélisation, il est utile de préciser le sens et la portée de l'outil d'évaluation, à savoir le modèle de simulation. Celui-ci est considéré comme une simplification de la représentation du réel, son degré de complexité étant fonction de l'objectif choisi (voir ci-dessus).

L'estimation des performances fournira un ensemble de valeurs : l'interprétation de celles-ci devra tenir compte des procédures mises en jeu et introduites au niveau du simulateur ; c'est par rapport au noyau des fonctions évaluées que les valeurs déduites seront significatives. De façon globale, la procédure se résume de cette façon : à la séquence de demandes de ressources introduite en input, le modèle va répondre par un output reprenant les performances obtenues par le système dans le cadre de la charge de travail donnée (Graham [3]). Le dernier point est important car il doit permettre à l'analyste de système d'apprécier l'efficacité du produit lorsque celui-ci est soumis à des input de nature différente.

Sur un plan fonctionnel, le contenu du modèle doit refléter les lignes de force qui s'inscrivent dans l'O.S. lui-même. Il n'est certes pas aisé de faire le choix entre l'indispensable et l'accessoire, étant donné la complexité toujours grande du produit à évaluer. Pour mener cette recherche à son terme, le concours du responsable intéressé par les résultats de l'estimation peut se révéler très utile ; en effet, il est en mesure de focaliser l'attention et de diriger les efforts de l'équipe d'évaluation sur l'une ou l'autre tendance du système d'exploitation primordiale à ses yeux. Ce qui n'implique pas nécessairement une ignorance totale et définitive des mécanismes laissés de côté dans une première étape de conception et de réalisation.

Un exemple illustrera mieux cette notion :

- le "Data Management" constitue une des fonctions majeures d'un système d'exploitation ; à ce titre ce serait un non-sens de vouloir le passer sous silence dans un modèle d'évaluation. Cependant, l'objectif fixé par les responsables peut ne pas se situer, dans un premier temps, au niveau d'une estimation détaillée de l'efficacité de la gestion des données. A titre exemplatif, l'optimisation de l'occupation d'un disque, ou la politique de gestion de buffers la mieux adaptée peut ne pas constituer un but à atteindre de façon prioritaire. Conséquemment, la conception du modèle de simulation considèrera la représentation du "Data Management" comme secondaire mais néanmoins non nulle. En effet, il existe certaines contraintes dont on devra tenir compte : sans entrer dans les détails de la gestion d'un disque ou d'une bande magnétique, la demande par un processus

d'éléments enregistrés en mémoire auxiliaire devra se traduire par une entrée-sortie. Dès lors, c'est au niveau du type d'informations recueillies lors de la simulation de cette opération que l'on introduira une représentation détaillée ou non de la gestion des données.

De tout ceci, on retient que les principales fonctions d'un système d'exploitation doivent pouvoir être prises en compte dans l'élaboration d'un modèle de simulation. Le degré de leur intervention à l'intérieur même de ce modèle sera très probablement différent, ceci à cause des objectifs fixés antérieurement, mais il est important de considérer l'O.S. comme formant un ensemble de parties intimement liées. En effet, il serait possible de réaliser de manière indépendante une étude de performances sur un algorithme de partage du CPU, sur une gestion de la mémoire centrale ou sur le déroulement des entrées-sorties, et de tirer ensuite des conclusions sur chaque recherche.

Mais alors, l'aspect vital qui serait négligé dans une telle évaluation est celui de l'interdépendance des différentes fonctions envisagées, et donc de l'impact qu'elles peuvent avoir l'une sur l'autre. En d'autres termes, le test "d'acceptation" du modèle réside dans le fait que ce modèle puisse fournir au responsable l'information relative aux performances du système d'exploitation considéré comme un tout (Kimbleton [10]).

c) La notion du temps. Un critère de distinction des simulateurs possibles est constitué par la présence ou l'absence d'une horloge au sein du modèle.

Dans la première hypothèse, une notion importante est évidemment celle du temps. Celui-ci intervient à deux niveaux :

1) Le temps "simulateur" est celui nécessaire à l'exécution du modèle de simulation.
2) Le temps "simulé" est celui que prendrait l'exécution d'un (de plusieurs) processus, celui-ci (ceux-ci) étant alors sous contrôle du système simulé.

On peut parler respectivement de "temps réel" et de "temps virtuel", en précisant qu'ils n'ont pas entre eux de rapport significatif ; à ce point de vue, le seul élément digne d'intérêt réside dans le fait qu'il faut essayer d'obtenir, pour une valeur donnée du temps simulateur, un temps simulé le plus grand possible. Nonobstant, cette obligation ne doit pas toujours être suivie de manière scrupuleuse : en effet, plus la tranche de temps affectée à l'exécution du modèle est grande, meilleure est la possibilité d'accumuler des informations sur le déroulement des processus simulés et sur l'efficacité du système. De nouveau, une solution de compromis devra intervenir.

d) Implications au niveau de la conception et de la réalisation du modèle.

On peut résumer ici ce qui va constituer les fondements d'une étude de simulation, telle qu'elle est conçue dans la suite.

- l'objet de cette étude étant l'évaluation d'un operating system, et la complexité de celui-ci étant supposée telle qu'il soit impossible de prendre tout en considération, il est nécessaire de dégager, au vu des objectifs choisis, un certain nombre de fonctions fondamentales autour

desquelles doit s'articuler la simulation.

- ces fonctions elles-mêmes sont obligées, à cause des spécifications du design, de tenir compte de nombreuses caractéristiques inhérentes aux éléments dont elles assurent le contrôle. Il n'est peut être pas opportun, dans un premier temps, d'introduire en simulation toutes ces caractéristiques. Il existera donc une phase de simplification des fonctions.
- lorsque le système réel travaille, il doit gérer toute une variété d'entités (englobant des processus, des ressources mémoire centrale ou auxiliaire, des fichiers, ...) qui à ce moment, existent effectivement. Lors du déroulement d'une simulation, elles sont seulement supposées exister, et ne sont présentes en fait que dans l'imagination du concepteur et du réalisateur. On est donc forcé d'opérer, non sur ces entités elles-mêmes, mais sur d'autres qui constituent uniquement une représentation des premières.

Enfin, d'un point de vue plus général, il est souhaitable de garder à la méthode de simulation un caractère ouvert, de façon à pouvoir insérer facilement et efficacement dans le modèle des fonctions du système autres que celles prévues dans une première étape.

- - - - -

1.4. RESTRICTIONS ET OBJECTIF FONDAMENTAL DE L'ETUDE

1.4.1. Restrictions apportées

Selon Lucas [2] , il existe deux types de simulation utilisés pour l'évaluation des performances d'un système :

a) le premier combine l'emploi des distributions de probabilités avec l'apparition d'une suite d'événements caractéristiques du fonctionnement du système.
b) le second fait usage des données obtenues de manière empirique.

En ce qui concerne l'optique de la suite de ce travail, une des restrictions apportées sera celle-ci : il ne sera pas fait appel (ou très peu) aux possibilités de simulation à partir de distributions théoriques, qui pourraient par exemple représenter les arrivées d'événements divers :

- entrée d'un nouveau processus dans le système,
- demande par un processus de la ressource CPU,
- demande par un processus d'une certaine capacité de mémoire (interne ou auxiliaire),
- suspension d'un processus,
- faute de segment,
- etc...

Une conséquence directe de cette prise de position sera l'absence de tout traitement mathématique des files d'attente générées au cours de l'exécution du modèle. Par contre, dans un souci d'approcher la réalité, l'usage de données dérivées de manière empirique sera prévu, ainsi d'ailleurs que le principe de la simulation par événements, qui sera précisé par la suite.

Un autre type de restriction apportée à cette étude concerne le moyen d'implémentation du modèle envisagé.

Sur ce point, la suite de ce travail ne donnera pas d'autres éléments que ceux qui vont suivre. On distingue généralement deux classes d'outils de réalisation d'un modèle de simulation : la première comprend tous les langages spécialisés en ce domaine (GPSS - SIMSCRIPT....) la deuxième est réservée aux autres langages qui, sans avoir été destinés spécialement à un tel usage, peuvent éventuellement faire l'objet du choix des responsables. Le but poursuivi ici n'étant pas de dégager avantages et inconvénients de l'une et l'autre classe, ces notions ne seront plus examinées dans la suite.

1.4.2. Objectif fondamental

L'objectif principal de ces quelques pages est le suivant : proposer par une méthode de simulation une approche du problème de l'évaluation des performances d'un système d'exploitation. Les fonctions primordiales de celui-ci telles que la gestion des processus et la gestion des ressources seront examinées plus en détail dans le cadre de la technique utilisée, en termes suivants : comment envisager la représentation des différentes entités, comment envisager le fonctionnement simulé du système de façon à rester fidèle à la réalité, et de manière à permettre au modèle de recueillir des informations en qualité et quantité suffisantes sur le problème posé.

Il est à noter que la discussion de résultats éventuels obtenus par un modèle implémenté et concernant tel ou tel operating system particulier n'est pas prévue dans cette étude.

- - - - -

1.5. SCHEMA GLOBAL DE L'APPROCHE PROPOSEE (1)

Le schéma figurant en page 24, peut, après un rapide examen de son contenu, être divisé en trois grandes parties qui vont être brièvement commentées ci-après.

Les chapitres suivants seront exclusivement consacrés à un développement plus détaillé des problèmes rencontrés lors de la conception d'un simulateur, ainsi qu'à une proposition de solution pour certains d'entre eux.

1.5.1. Input du modèle

La volonté de se rapprocher le plus près possible du réel pour estimer les performances d'un operating system existant mais pour lequel on ne dispose pas encore de résultats statistiques élaborés, débouche naturellement sur une politique de mesures appropriées en vue de pouvoir fournir au simulateur un input valable.

On peut par exemple distinguer trois stades dans la recherche de cet input :

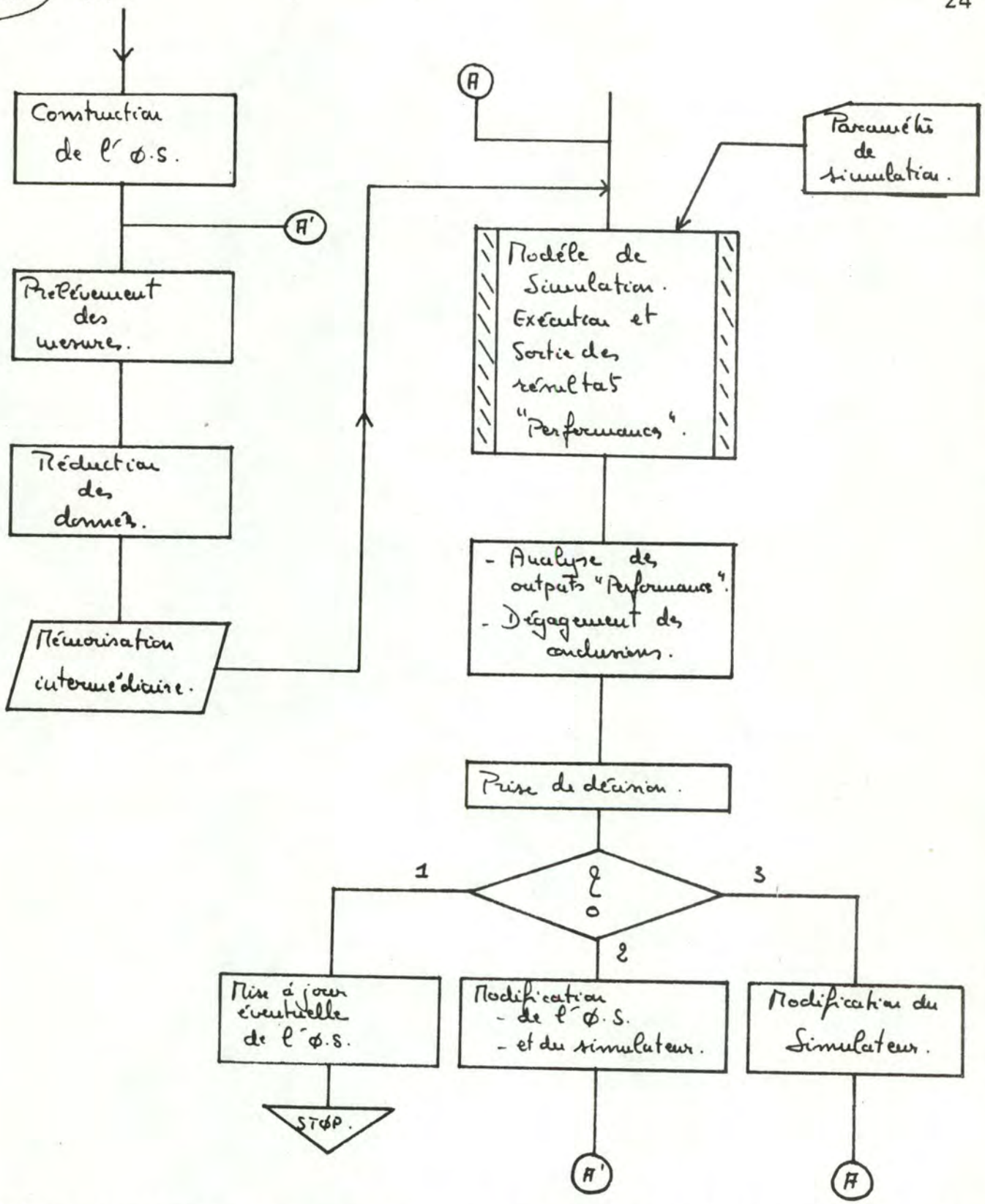
a) un stade de collecte des données (par monitoring software ou hardware) dont la contribution est jugée nécessaire lors du déroulement de la simulation.

b) un stade de réduction de données, destiné à mettre sous forme cohérente et utilisable les informations recueillies lors de l'étape précédente.

---

(1) Ce schéma n'a pas la prétention d'être le meilleur ou le plus complet. Il constitue seulement une approche du problème de l'évaluation des performances et c'est à ce titre seulement qu'il doit être considéré.

SCHEMA (1)



(1) La branche droite du test pourrait, de façon à ce que le schéma soit plus complet, présenter le choix suivant :

- la modification du simulateur dans le cadre des fonctions déjà représentées,
- ou la représentation d'autres fonctions.

c) un stade de mémorisation du résultat obtenu en attendant l'introduction de celui-ci en simulation proprement dite.

Il est à noter que Martin [11] et Noetzel [6] donnent à ce propos une information plus dense et digne d'une attention particulière.

1.5.2. Déroulement de la simulation

Etant donné :

a) l'input obtenu en effectuant des mesures préalables,

b) les paramètres spécifiques au déroulement de la simulation,

c) les options prises dans la conception et la réalisation du modèle lui-même (supposé testé et mis au point),

on peut alors réaliser l'exploitation systématique du simulateur en soumettant à celui-ci diverses charges de travail. Elles peuvent apporter une variété de renseignements à différents niveaux : selon la nature des ressources les plus demandées, la requête plus ou moins importante en quantité de mémoire centrale, le nombre et la dimension des fichiers traités, etc..., le fonctionnement du système stimulé présentera des variations de comportement, et produira donc un ensemble d'informations relatives aux performances qu'il faut espérer le plus riche possible.

1.5.3. Analyse des outputs et décisions conséquentes

La phase ultime de l'évaluation est évidemment pour les responsables du système la plus intéressante. De façon globale, elle peut être envisagée comme suit :

a) une analyse approfondie des résultats produits par la simulation doit permettre de dégager des normes de comportement du système évalué.

b) en fonction des conclusions de cette analyse, la poursuite de l'amélioration de l'operating system peut prendre trois directions différentes :

1/ on décide d'arrêter l'évaluation et on met à jour éventuellement la version de l'O.S.

2/ une modification de l'O.S. et par conséquent du simulateur (pour que ce dernier soit conforme au produit évalué) est décidée et réalisée.

3/ un changement au niveau du modèle d'évaluation seulement est entrepris.

Il va de soi que ces opérations sont très délicates et qu'une amélioration d'un système d'exploitation risque d'entraîner des coûts non negligeables. Il est donc primordial pour un responsable d'engager ces procédures avec prudence.

*

* *

C H A P I T R E II :

INPUT DU MODELE D'EVALUATION

2.1. Objectifs choisis et type de données conséquent.

2.2. Workload et paramètres de configuration.

2.3. Paramètres de simulation.

2.4. Note sur les caractéristiques du software simulé.

2.1. OBJECTIFS CHOISIS ET TYPE DE DONNEES CONSEQUENT

2.1.1. Objectif choisi.

Avant d'aborder plus en détail l'examen des différents problèmes à résoudre pour concevoir un simulateur, il n'est pas inutile de rappeler brièvement l'optique dans laquelle celui-ci est envisagé. Il s'agit d'aboutir à la possibilité de construire un modèle d'estimation des performances d'un operating system existant qui permette, dans le cadre de charges de travail variées, de dégager des normes de comportement du produit sujet à l'évaluation. Le propos de ce chapitre consiste à déterminer le mieux possible les informations nécessaires pour constituer l'input d'un simulateur.

2.1.2. Types de données à introduire.

Lorsque l'utilisateur fait appel aux services d'un computer system, il soumet à celui-ci une suite de programmes à exécuter qui représentent l'automatisation de certaines fonctions. Le résultat attendu s'exprime sous forme de fichiers dont le contenu est utilisable par l'homme. La question est la suivante : comment appréhender toute la procédure de traitement et son environnement de façon à pouvoir estimer l'efficacité du système d'exploitation ? Pour ce faire, on doit déporter pour l'instant son attention du niveau "problème" (naturel pour l'utilisateur) au niveau "machine" (naturel pour l'O.S.). En effet, c'est à ce dernier degré qui sera celui de l'exécution (1), que l'on fait usage des ressources hardware et software.

---

(1) les problèmes relatifs à la compilation (ou à l'assemblage), à l'édition des liens et au chargement ne seront pas examinés.

Les informations de base à recueillir seront par conséquent formulées dans les termes suivants :

- de quelle ressource le processus a-t-il besoin ?
- quand et pendant combien de temps en a-t-il besoin ?
- en quelle quantité ?
- quel est ce processus ?

L'ensemble des réponses à ces questions constitue le point de départ de la construction de l'input.
Th. Williams [12] parle de trois éléments majeurs à introduire dans un simulateur : le workload, les entités hardware, les entités software. Ce sont les deux premiers qui sont concernés par les questions posées ci-dessus et ce sont eux qui vont fournir à la simulation une base de travail. En d'autres termes, il est nécessaire avant tout traitement simulé de disposer des informations suivantes :

a) les caractéristiques des différents dispositifs hardware principaux de la configuration.
b) une schématisation du workload apte à représenter le plus fidèlement possible les appels de ressources qu'aurait provoqué, dans la réalité, l'exécution du job de l'utilisateur.

Il est à remarquer le sens de la démarche suivie : de l'aspect "problème" envisagé au début, on passe à l'aspect "machine" pour revenir ensuite au premier considéré.

On satisfait en somme à deux contraintes imposées par le problème de l'évaluation des performances tel qu'il est posé ici :

1) examiner le fonctionnement du système en se plaçant au niveau du travail qu'il réalise effectivement, c'est-à-dire l'allocation de ressources,
2) tout en gardant la structure de la charge de travail réelle par une schématisation appropriée

du workload introduite en input.

Une question vient de suite à l'esprit lorsqu'on a pris connaissance des exigences du modèle sur les données qu'il va traiter : comment va-t-on pouvoir disposer de celles-ci ? A ce sujet on distingue plusieurs cas :

a) les caractéristiques hardware de la configuration sont connues.
b) de même, celles des fichiers manipulés par les différents programmes d'application.
c) par contre, en ce qui concerne l'identification des processus et les temps d'utilisation des ressources qui leur sont allouées, il va falloir recourir à des moyens de mesure.

De façon globale, les procédures de mesure, qu'elles soient exécutées par monitoring hardware ou software, obéissent au schéma présenté en fig. 2.1.

Lors de l'exploitation d'une application quelconque, des mesures sont effectuées sur les conditions dans lesquelles l'input, le traitement et les outputs relatifs à ces applications sont réalisés.

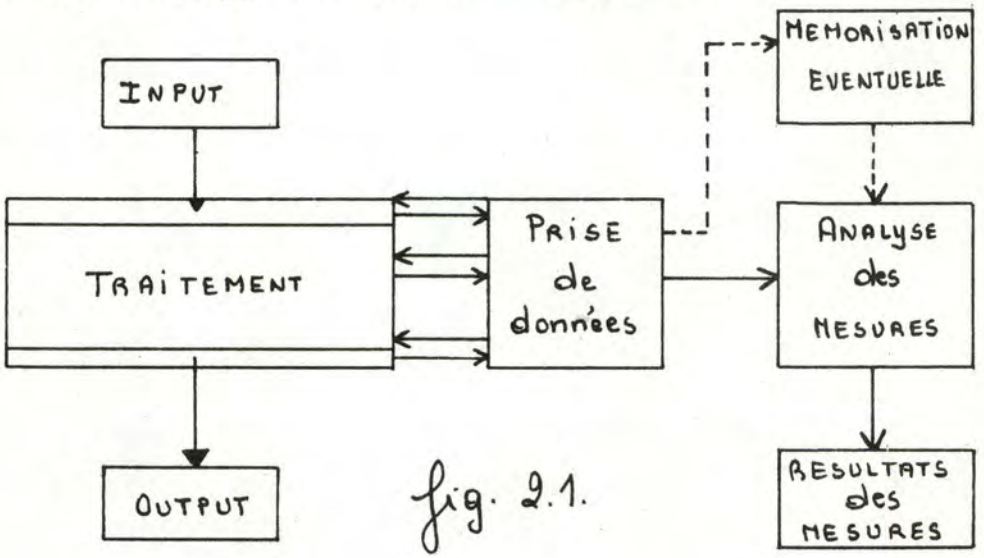


fig. 2.1.

Si le moniteur n'est pas suffisamment rapide pour pouvoir analyser les renseignements de suite, il s'agit uniquement là d'une collecte d'informations. Celles-ci, après une mémorisation intermédiaire, subiront alors une analyse différée avant de livrer aux responsables intéressés les caractéristiques résultant du fonctionnement du système d'exploitation.

Etant donné que le but de ce travail n'est pas de dégager une méthode ou une philosophie des mesures dans le cadre de l'évaluation des performances d'un O.S., cet aspect des choses ne sera pas développé plus longuement ici. Gothieb [5] , Lucas [2] et Calingaert [4] donnent à ce sujet quelques éclaircissements supplémentaires.

2.1.3. Un exemple de réalisation.

Il pourrait sembler à première vue que l'obtention de toutes ces caractéristiques (configuration, charge de travail, ...) et leur utilisation en simulation soient difficiles voire impossibles. En fait, il n'en est rien (Noetzel [6]) et des représentations des différentes tâches introduites par l'utilisateur peuvent être déduites et utilisées à bon escient.

A titre d'exemple, une schématisation de la charge de travail fut réalisée lors des études d'évaluation de systèmes existants qu'a effectué l'équipe PRESTE-SEMA par un procédé de comparaison des performances.

L'outil (1) dont il est fait usage est un modèle de simulation qui présente les particularités suivantes.

A partir de :

1) la définition de la configuration à tester,
2) la définition d'une ou de plusieurs chaînes de traitement d'un utilisateur,

(1) Un autre exemple est constitué par le modèle CASE, proposé par TESDATA.

3) la définition de l'environnement d'exploitation (contraintes spécifiques, ...),

le modèle livre en sortie (1) un ensemble de valeurs statistiques caractérisant l'utilisation des ressources selon la configuration et les fonctions remplies dans la chaîne du traitement.

- - - - -

(1) Les résultats des expériences sont mentionnés dans les parutions de 01 - INFORMATIQUE des mois suivants : septembre 72 - octobre 72 - novembre 72 - janvier 73 - février 73 - avril 73

2.2. WORKLOAD ET PARAMETRES DE CONFIGURATION

En vertu de l'optique choisie, il s'agit de rassembler les caractéristiques d'input dont la présence est jugée nécessaire à l'intérieur du modèle de simulation. Ces caractéristiques couvrent le domaine suivant :

a) ce que le système connaît à priori, c'est-à-dire l'état et la puissance de la configuration.

b) ce qu'il connaît au moment de l'exploitation, c'est-à-dire tout ce qui constitue la charge de travail ou y est relatif.

En fonction de cette distinction, on peut envisager de la façon suivante la découpe des inputs du simulateur :

a) les ressources hardware.

b) les éléments appartenant ou liés au workload.
   - les chaînes de traitement
   - les différents fichiers utilisés
   - si le cas se présente, une division particulière de l'espace d'adresses, dont la responsabilité échoit à l'utilisateur
     exemple : une mémoire virtuelle segmentée.

Il est utile de noter ici que pour une mémoire virtuelle pagée, la caractéristique "taille de la page", bien qu'elle ne dépende pas de l'initiative de l'utilisateur, intervient sous la forme d'une gamme de valeurs de départ ; l'utilisateur de celles-ci en simulation doit permettre de sélectionner la (les) meilleure (s) option (s).

Pourquoi opérer la distinction entre d'une part, l'aspect "hardware pur" et d'autre part l'aspect "utilisation du hardware" ? La réponse se trouve déjà plus ou moins dans la formulation de la question :

- la première notion se rapporte à un ensemble d'entités qui généralement reste inchangé pour une série d'exploitations du simulateur,

- tandis que la deuxième couvre un domaine qui lui, peut se révéler très varié, selon l'application que l'on veut réaliser.

Il ne faut pas oublier en effet que le but cherché reste l'évaluation des performances obtenues par un système d'exploitation lorsqu'on soumet à celui-ci des charges de travail de nature et de quantité variables, mais où l'on considère, dans un premier temps du moins, que la configuration sous-jacente reste fixe.

On peut résumer ce qui précède par le schéma suivant :

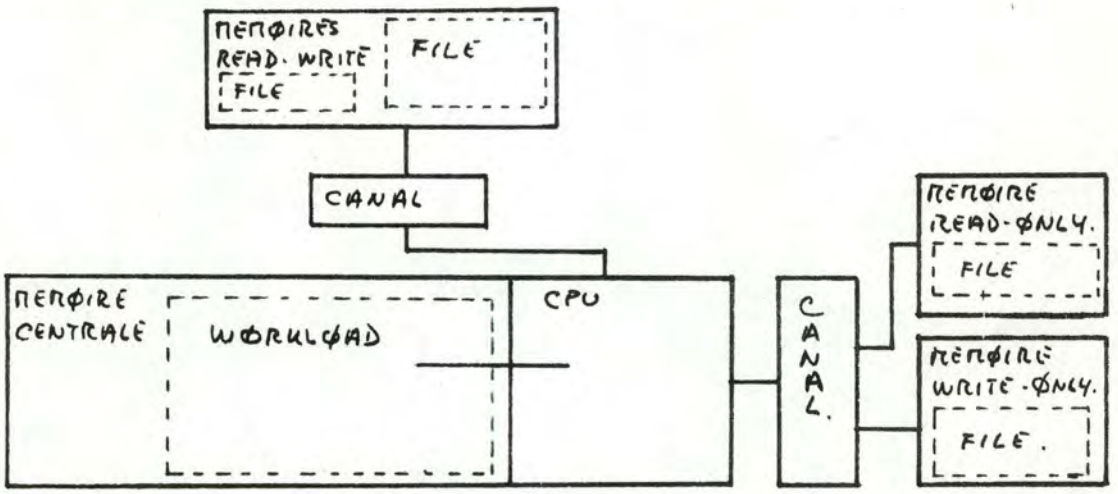


fig. 2.2.

On reprendra les éléments de cette figure (qui prévoit une configuration assez simple) et à chacun d'eux on associera les caractéristiques dont la contribution apparaît comme intéressante en simulation.

2.2.1. Les ressources hardware

a) le CPU : il est pris en compte au niveau des temps simulés que l'on accorde aux différentes phases logiques du processus ; cette notion de phase logique sera précisée ultérieurement (cf. infra p. 58)

b) les canaux : pour pouvoir simuler les entrées-sorties il sera nécessaire de disposer chaque fois des renseignements suivants : quel est le canal associé au dispositif I/O sur lequel se trouve le fichier

concerné ? Le canal sollicité est-il libre ou occupé ?

c) la mémoire centrale : la seule caractéristique intéressante à relever à ce niveau est incontestablement la capacité de la mémoire.

d) les mémoires auxiliaires : à la différence de la mémoire centrale, chaque unité de mémoire auxiliaire nécessite une définition plus détaillée. La raison en est simple : étant donné que d'une part, l'examen des performances est pensé en termes de place et de temps, que d'autre part l'utilisation de ces mémoires auxiliaires va engendrer une procédure tout à fait différente de ce qui se passe lorsqu'on veut accéder à un endroit de la mémoire centrale (où les opérations sont quasi immédiates), il est naturel, pour être fidèle à la réalité, de reporter en simulation les éléments qui interviennent significativement dans l'utilisation d'une mémoire secondaire.

Avant d'envisager le détail des différentes caractéristiques, il est utile de préciser la notion suivante : les mémoires auxiliaires sont ici considérées en termes de dispositif hardware permettant un stockage de l'information dans une certaine quantité, et éventuellement une recherche des données mémorisées qui nécessitera l'écoulement d'un quantum de temps dont la valeur est la combinaison des différents temps élémentaires. Il est clair que ceci constitue une approche générale ; par conséquent, selon le dispositif qu'on examine, on sera naturellement amené à prendre en considération tout ou partie des éléments qui constituent son fonctionnement particulier.

On distingue les cas suivants :

1) <u>les dispositifs à vocation unique</u> : on désigne par cette appellation ceux sur lesquels il n'est possible d'exécuter qu'une opération de nature unique,

à savoir, ou une mémorisation, ou une recherche d'information.

Ce sont :

a) le lecteur de cartes : la caractéristique essentielle relevée est relative au temps, et l'on néglige la capacité, étant donné que celle-ci, en principe du moins, est pratiquement illimitée.
On retient donc, outre la référence du dispositif, le temps nécessaire pour lire un enregistrement physique, c'est-à-dire une carte.

b) le perforateur de cartes : en vertu de la même raison que celle énoncée ci-dessus, on retient le temps nécessaire pour sortir un enregistrement physique, ainsi que la référence du dispositif.

c) l'imprimante : de la même façon, on considère le temps nécessaire pour imprimer une ligne, et la référence voulue.

En ce qui concerne cette catégorie, il apparaît comme inutile, étant donné la dimension fixe des enregistrements physiques et l'absence de problèmes d'adressage, de descendre dans le détail de la représentation des mécanismes sous-jacents.

2) les dispositifs à vocation double : contrairement aux dispositifs examinés ci-dessus, ceux-ci présentent les particularités suivantes :

- la possibilité de communiquer dans les deux sens avec l'organe central de mémorisation.
- l'existence dans certains cas de problèmes d'adressage.

- l'absence de longueur fixée pour les enregistrements physiques.

L'impact immédiat de ces deux dernières notions s'exprime comme suit : il est désormais nécessaire pour tenir compte fidèlement des mécanismes inhérents à ces dispositifs, de décomposer en leurs phases élémentaires les temps requis pour accéder à l'information mémorisée ou pour stocker celle-ci où elle doit l'être.

On distingue :

a) le tambour magnétique. Les caractéristiques principales en sont :

- la référence
- la capacité
- le temps de rotation
- la vitesse de transfert des caractères

b) le dérouleur de bande magnétique : on relève les notions suivantes :

- la référence du dispositif
- le temps passé avant la décélération
- le temps de décélération
- le temps d'accélération
- la vitesse de transfert des caractères
- la longueur des espaces inter-blocs.

Schématiquement, ces différents éléments peuvent être représentés de cette manière :

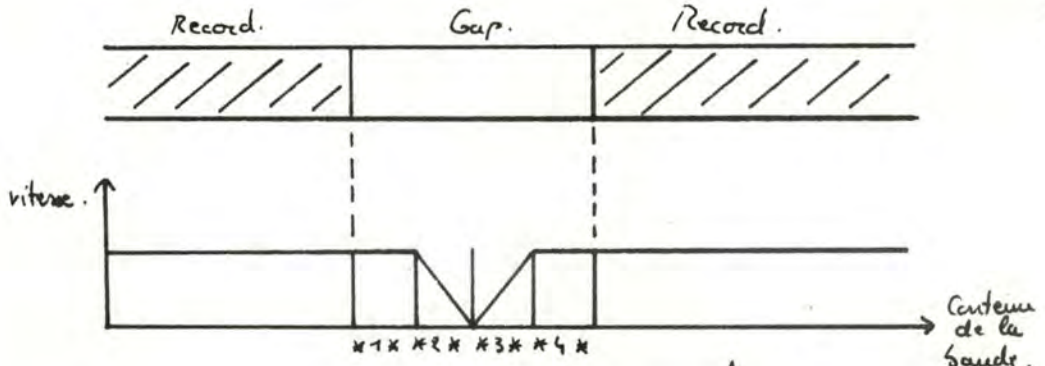


fig. 2.3.

Sur cette figure, les intervalles chiffrés sont affectés de la signification suivante :

l'intervalle *1* correspond au temps passé avant décélération.

l'intervalle *2* correspond au temps passé pendant la décélération.

l'intervalle *3* correspond au temps passé pendant l'accélération.

l'intervalle *4* correspond au temps de stabilisation.

c) le discpack. Les caractéristiques principales en sont :

- la référence
- le temps de rotation
- le temps de positionnement (minimum, moyen, maximum) du bras
- le nombre de caractères par piste
- le nombre de pistes par cylindre
- le nombre de cylindres par unité de disques.

2.2.2. Les caractéristiques des fichiers

Après avoir considéré le cas des ressources hardware, passons à l'analyse de l'utilisation de ces ressources.

Si l'on se reporte à la fig. de la page 34, on constate qu'il reste des éléments de deux natures à prendre en considération, le tout répondant au critère d'utilisation

de ressources :

- des données à traiter, qui en l'occurrence existent sous la forme de fichiers.
- des procédures de traitement elles-mêmes, ou le work-load du système.

La seconde de ces natures étant examinée en 2.2.3., envisageons maintenant le problème des fichiers.

De même que le simulateur ne connaît pas de ressources hardware proprement dites, il n'a pas non plus connaissance des fichiers réels relatifs aux applications automatisées. Il s'avère donc nécessaire de lui fournir un outil qui lui permettra d'éxécuter son travail de la manière voulue, c'est-à-dire de "faire comme si" il disposait, à l'instar de l'operating system, de tous les data sets nécessaires aux chaînes de traitement prévues.

Dans ce but, on lui donne, comme pour les ressources hardware, un ensemble de "descripteurs" des fichiers existants dans le cadre des applications réelles.

Ces descriptions de fichiers doivent contenir :

- une référence de fichier.
- un indicateur de la méthode de constitution du fichier si celui-ci n'existe pas encore.
- un indicateur de la méthode d'accès.
- la référence du dispositif de mémorisation sur lequel on suppose le fichier.
- la longueur de chaque enregistrement physique (si on suppose une longueur fixe).
- si on travaille en longueur variable, et que l'on connaît la plage de variation de cette longueur : une indication de la longueur maximum, et de la longueur minimum.
- si on prévoit une gestion simulée d'allocation de buffers : le nombre de buffers nécessaires pour le fichier.

- si le fichier est mémorisé sur disque : l'adresse de début du fichier.
- le nombre d'enregistrements dans le fichier.

2.2.3. Les caractéristiques de la charge de travail

Après avoir passé en revue les principales ressources hardware et les ensembles des données à traiter, il reste à prendre en compte ce qui finalement constitue l'élément moteur de tout ceci, c'est-à-dire les procédures de traitement.

Du point de vue qui nous intéresse, elles sont considérées essentiellement comme utilisation de différentes ressources, à différents niveaux (Wood - Forman [14]). Le déroulement de la simulation retient uniquement cet aspect, ce qui n'empêche nullement le responsable, au stade de l'interprétation et de l'analyse des résultats, d'effectuer son travail en fonction de l'input présenté au simulateur. En réalité, la structure de la charge de travail et la nature des applications qu'elle représente constituent un élément important dans l'évaluation des performances d'un operating system ; en effet, celles-ci sont toujours définies en tenant compte d'une certaine relativité.

En d'autres termes, le problème est de trouver, à partir des chaînes de traitement soumises continuellement au contrôle du système, une représentation fidèle, à la fois sur les plans structurel et quantitatif, de la séquence des demandes de ressources diverses adressées au système d'exploitation.

De manière globale, et toujours selon le schéma de la page 34 , on peut caractériser le workload selon les appels qu'il provoque aux ressources de type suivant :

- mémoire principale.
- organe central de traitement.
- dispositifs auxiliaires de mémorisation.
- interfaces mémoire principale - mémoires secondaires

Au vu de cette découpe, on est en droit de se poser cette question : dans quelles circonstances ces différentes ressources sont-elles l'objet d'une requête venant de la charge de travail ?

a) en ce qui concerne la mémoire principale, il s'agit là d'un problème d'allocation dont la relation appartient à l'O.S.
b) l'organe central de traitement est requis chaque fois qu'une instruction est en passe d'être exécutée.
c) quant aux mémoires auxiliaires, elles sont sollicitées à chaque demande (ou envoi) d'information en provenance de la mémoire centrale.
d) les interfaces d'entrée - sortie enfin, sont concernés lorsque les dispositifs secondaires de mémorisation le sont également.

En conséquence de ceci, il apparaît logique d'opérer la distinction suivante :

a) ce qui se trouve sous le contrôle du système (ex. : allocation de mémoire principale) ne relève pas de l'utilisateur et n'est donc pas visible directement par le workload.
b) pour le reste, il est possible d'intégrer dans l'input du modèle ce qui en fait constitue naturellement la charge de travail, à savoir la demande du CPU ainsi que celles des entrées - sorties.

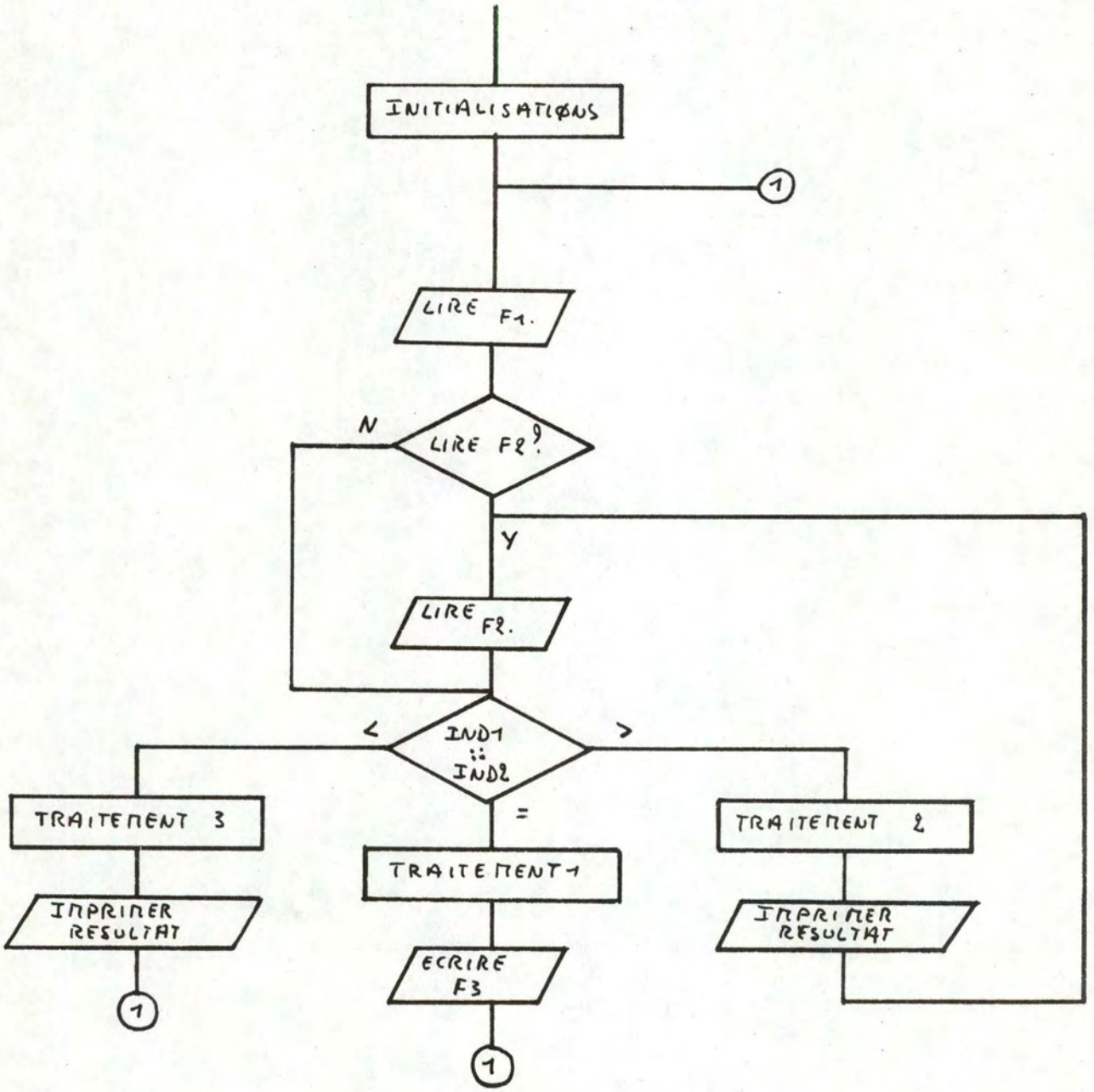
En attendant de poursuivre plus avant l'étude de ce paragraphe, un exemple aidera, si besoin en est, à clarifier les notions envisagées et à les compléter.

Exemple
.....

Soit à réaliser le traitement dont les caractéristiques sont les suivantes :

- on dispose de deux fichiers sur bande, que l'on suppose indépendants et mis à jour dans d'autres applications.
- ces fichiers contiennent des enregistrements dont le contenu est de nature différente, mais qui possèdent cependant un indicatif dont la plage de variation est identique.
- dans le cas de correspondance d'indicatifs, on effectue un traitement approprié et on stocke l'enregistrement résultant sur un troisième fichier bande.
- sinon, on réalise un autre type de traitement, et on imprime son résultat.

De façon globale, et sans examiner ici le problème posé par la fin d'un fichier, on peut établir comme suit l'organigramme logique de fonctionnement.


INITIALISATIONS
1
LIRE F1.
N
LIRE F2 ?
Y
LIRE F2.
<
IND1 :: IND2
>
=
TRAITEMENT 3
TRAITEMENT 1
TRAITEMENT 2
IMPRIMER RESULTAT
ECRIRE F3
IMPRIMER RESULTAT
1
1

Laissons maintenant de côté l'aspect "problème" pour accorder notre attention à celui qui doit guider la démarche dans le processus de schématisation du workload, à savoir l'aspect "utilisation de ressources".

On constate :

- un traitement interne, c'est-à-dire une demande de CPU (et conséquemment de mémoire centrale) lors de l'exécution des blocs : initialisations - traitement 1 - traitement 2 - traitement 3.
- plusieurs demandes d'entrées - sorties, c'est-à-dire une utilisation des ressources canal et dispositifs de mémorisation secondaires lors des opérations : lecture du fichier 1 - lecture du fichier 2 - écriture du fichier 3 - impression de résultats.

En conséquence, pour une application soumise à l'exécution sous contrôle de l'O.S., il sera nécessaire, en raison de l'optique choisie, de préciser au simulateur :

- d'une part la structure du workload telle qu'elle apparaît sur l'organigramme logique.
- d'autre part les paramètres d'ordre quantitatif jugés primordiaux tels que :
  - la durée d'exécution des différents blocs, qu'ils concernent le CPU ou les processeurs d'entrées - sorties.
  - les références des ensembles structurés de données concernées par les opérations I /O.

Il est à remarquer que la nécessité de fournir au modèle l'ossature du workload demande d'introduire dans celle-ci non seulement les stades principaux d'utilisation d'un processeur, mais également les moyens de pouvoir choisir entre les différents chemins existants dans le graphe logique. La prévision des points de débranchement ainsi que de rebranchement se révèle donc comme fondamentale.

Si l'on résume brièvement les éléments examinés jusqu'à présent, et dont la participation à l'input du simulateur est considérée comme vitale, on obtient la liste suivante :

- les ressources hardware
- les caractéristiques des fichiers
- la schématisation de la charge de travail

Le point de vue considéré dans ces pages résidait uniquement dans la détermination de la nature de l'information à introduire au niveau des données traitées par le modèle d'évaluation. Le problème de l'implémentation de ces données constitue un autre aspect d'une étude d'estimation des performances d'un operating system ; et bien que, en toute évidence, il doive entrer en ligne de compte lors de la réalisation du modèle, il ne sera pas abordé de manière directe tout au moins, dans la suite de ce travail.

2.2.4. Le cas d'une gestion particulière de mémoire

Il peut paraître incorrect d'insérer, dans un chapitre consacré à l'input d'un simulateur, des notions concernant une méthode de gestion de la mémoire, étant donné que d'une part, les procédures sous-jacentes à une telle gestion relèvent du domaine de l'operating system, et que d'autre part, l'option choisie dans la construction de l'input désiré a pour but de répertorier l'ensemble des informations nécessaires au bon fonctionnement du modèle, mais externes à toute procédure système.
Ce principe est respecté tout au long des pages qui précèdent ; en effet, qu'il s'agisse des ressources hardware, des caractéristiques des fichiers ou des composantes principales du workload, tout ceci recouvre un ensemble d'éléments qui sont utilisés plus ou moins directement par les routines de l'O.S., sans toutefois en constituer la substance même. Il n'est nullement

question dans ce paragraphe de déroger à cette règle, et les notions qui vont suivre doivent, à l'instar de celles énoncées depuis le début du chapitre,aider à la constitution des variables d'utilisation du simulateur, le noyau de celui-ci étant réservé à l'implémentation des mécanismes sujets à l'évaluation.

Le type de gestion de mémoire envisagé est celui d'une mémoire virtuelle segmentée.

De la même façon que pour les éléments examinés précédemment dans ce chapitre, on peut affimer cette fois encore qu'il existe des caractéristiques, dues à la segmentation, susceptibles de figurer dans l'input du simulateur, étant donné que l'utilisateur définit lui-même certaines valeurs associées aux segments. Il est donc normal que cette intervention de l'utilisateur trouve son correspondant au niveau de la simulation.

Concrètement, ceci peut être réalisé en complétant l'input du modèle par les notions suivantes :

- taille du segment,
- ce segment est-il relogeable ou pas ?
- peut-on lui appliquer le swapping ?
- est-il privé ou partagé ?
- quel est le contenu du working-set associé au processus simulé ?
- etc...

Ceci n'est évidemment possible qu'en fonction d'une découpe logique de la mémoire ; il va de soi que dans le contexte d'une mémoire virtuelle pagée, il ne serait plus concevable d'agir de la même façon : en effet, contrairement au segment qui lui, est "visible" par l'utilisateur, la page, elle, dépend totalement, quant à sa dimension, d'une option appartenant au domaine du concepteur de système.

Ceci termine le développement consacré aux variables d'utilisation du simulateur ; de manière globale, il essaie de cerner l'ensemble des éléments qui, des caractéristiques de la configuration à celles de la charge de travail, agissent sur le comportement du système et risquent d'orienter sensiblement les performances de celui-ci.

- - - - -

2.3. PARAMETRES DE SIMULATION

En dehors des variables d'utilisation, mentionnées lors des sous-chapitres précédents, on peut définir un certain nombre de valeurs d'un autre type, qui ne seraient plus liées directement ni au problème posé ni à la configuration utilisée, mais bien à l'aspect de simulation lui-même. Les paramètres de simulation, étant donné leur nature même, se prêtent moins bien à une approche systématique et dépendent surtout de la façon dont on veut utiliser le modèle d'estimation. En d'autres termes, on pourrait justifier leur existence par le fait suivant : au-delà du système évalué, de la configuration sur laquelle il fonctionne et de la charge de travail qui lui est soumise, il faut prendre en compte le contexte dans lequel se trouve le simulateur, et fixer à ce contexte un ensemble de valeurs sur base desquelles le modèle d'évaluation doit être exécuté.

A titre exemplatif uniquement, on peut citer :

- l'espace de temps que l'on veut simuler.
- des critères de constitution (à l'exécution) de statistiques sur les files d'attente.
- des critères d'impression des résultats de ces statistiques.
- etc...

- - - - -

2.4. NOTE SUR LES CARACTERISTIQUES DU SOFTWARE SIMULE

On pourrait se demander pourquoi on n'introduit pas dans le modèle les caractéristiques du software en même temps que celles de la configuration et du workload. Une telle attitude aurait alors la signification suivante :

- le fait de pouvoir insérer dans l'input lui-même les propriétés et mécanismes fondamentaux du software (sous quelque forme que ce soit) impliquerait automatiquement une variabilité potentielle très élevée de ces caractéristiques et par conséquent, une tentation très forte et immédiate pour l'analyste de système à se servir de cette facilité.
- par conséquent, dans l'hypothèse d'un tel contexte, le cadre que l'on s'est fixé au début de l'étude, à savoir une estimation des performances d'un operating system existant, est bouleversé de manière à tenir compte de la possibilité d'évaluer l'efficacité de plusieurs systèmes d'exploitation. On débouche alors dès la première exécution sur un examen de type comparatif.

Une telle optique présente évidemment un avantage considérable en ce sens qu'elle permet pratiquement l'évaluation "à la carte" d'un certain nombre de systèmes auxquels on pourrait soumettre des workloads de même nature ou éventuellement de nature différente si la possibilité en est prévue.

Dans cet ordre d'idées certaines sociétés américaines ont conçu des modèles (ex. : la société COMRESS a élaboré le modèle SCERT) répondant à de telles exigences (Huesmann and Goldberg [1] ; [15]).

Néanmoins, ce n'est pas notre propos ici d'envisager une telle direction. Il n'est pas exclu cependant de prévoir à un certain stade d'exploitation du modèle un processus de comparaison; toutefois, celui-ci serait engagé dans un sens différent.

En effet, si l'on se réfère au schéma global présenté à la fin du chapitre I, il se peut qu'après un certain nombre d'utilisation du simulateur, l'analyse des résultats obtenus décèle un ou plusieurs points faibles dans les mécanismes simulés.
Il serait alors opportun de modifier en conséquence le modèle d'évaluation de façon que, lors des exploitations ultérieures, les changements apportés aient un effet bénéfique en ce qui concerne les performances. C'est dans cette optique qu'un procédé de comparaison de versions différentes d'un même système pourrait être retenu.

*

* *

C H A P I T R E III :

DEROULEMENT DE LA SIMULATION

3.1. Caractéristiques de la vie d'un processus.

3.2. Gestion simulée des processus.

3.3. Gestion simulée des ressources.

3.1. CARACTERISTIQUES DE LA VIE D'UN PROCESSUS

3.1.1. Processus et notion d'événement

Sans toutefois prétendre donner la meilleure approche de la notion de processus, on peut définir celui-ci comme une entité caractérisée comme suit :

a) elle s'identifie ou résulte de l'utilisation d'un processeur ;
b) elle se trouve dans un seul des 3 états suivants à la fois :
   - actif : le processeur utilise effectivement une ressource.
   - prêt : le processeur est prêt à utiliser une ressource ; toutefois, la disponibilité de celle-ci ne lui est pas assurée au moment où il se trouve dans cet état.
   - suspendu : le processeur est bloqué ; il se trouve en attente d'un signal externe qui lui permettra de passer dans l'état "prêt".
c) elle est "contrôlée" au sens large par l'operating system.

Cette définition a le mérite de sous-entendre l'aspect fondamental de "séquence d'utilisation de ressources" sur lequel est basée la simulation dans le cadre de ce travail. Cet aspect, déjà introduit dans le chapitre II à propos de la construction de l'input du modèle, sera complété dans les lignes qui suivent.

Dans l'optique "étude de comportement du système", il est nécessaire d'oublier un peu la position de l'utilisateur, qui se définit en termes de travaux (jobs) à soumettre au traitement automatique, pour adopter, si l'on peut s'exprimer de cette manière, la "façon de raisonner" propre au système. Qu'il y ait en vie un ou plusieurs processus simultanés, l'O.S. gère le fonctionnement de ceux-ci selon des règles bien établies :

il connaît à chaque instant leur état, les ressources qu'ils occupent, et décide de l'enchaînement de leurs états en fonction de conditions bien précises dont l'occurrence constitue ce que l'on appelle un "événement".

Au niveau du système (1), un événement survient chaque fois que se produit un changement d'état d'un processus sous contrôle du système (Graham [3]). Ce changement d'état peut être caractérisé par : (fig. 3.1.)

- l'abandon d'une ressource précédemment occupée par le processus {1} .
- la demande d'une ressource occupée à ce moment par un autre processus que le demandeur, et donc non encore disponible {2} .
- l'occupation d'une ressource par le processus {3} .
- l'arrivée d'un signal externe au processus, et qui rend celui-ci prêt à occuper une ressource {4} .

Pour employer un terme imagé, on peut dire que la photographie du système global est modifiée chaque fois que se produit, pour un processus quelconque, un changement d'état. En d'autres termes, un événement est la concrétisation d'une transition survenue dans la vie d'un processus, et il est nécessaire, pour pouvoir évaluer l'efficacité de l'O.S., d'en tenir compte de façon précise.

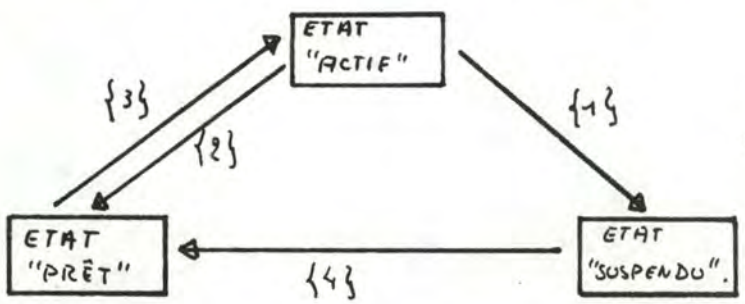


fig. 3.1.

(1) on ne considère pas encore ici la notion d'événement telle qu'elle apparaîtra au niveau de la simulation.

Si l'on se replace à un point de vue plus global, on constate que le système d'exploitation gère des séquences d'utilisations de chaque ressource, étant entendu qu'à une séquence ne correspond pas nécessairement un processus, si l'on considère une séquence comme une suite d'utilisations d'une ressource particulière.

Etant donné deux types de ressources principales (le processeur central et les dispositifs liés aux entrées-sorties) on peut établir en toute généralité un schéma (fig. 3.2.) relatif au flux des processus ; ce schéma est une adaptation de celui que donne Graham [3] pour un job-flow et contient les phases suivantes :

a) une phase d'initialisation : avant de lancer un processus, le système doit savoir qu'il existe une possibilité de le faire et prendre en conséquence les actions voulues.

b) avant de commencer (ou de continuer) à vivre, le processus doit recevoir du système le "feu vert" concernant une occupation du CPU ; dans le cas contraire, le processus est placé en attente de la ressource.

c) lorsque c'est possible, le processus occupe le CPU ; la fin de cette phase peut être déterminée par diverses règles ou conditions : parmi elles se trouve le terme de l'éxécution du processus et par conséquent la mort de celui-ci, ou la demande d'une entrée - sortie.

d) si cette dernière hypothèse est vérifiée, il est nécessaire de s'assurer de la disponibilité des éléments suivants :
   - le dispositif d'I / O voulu
   - et le canal qui en gère l'accès.

   Dans le cas d'une occupation de l'un ou l'autre dispositif, le processus d'entrée - sortie est placé en attente dans la file adéquate.

e) au moment où l'opération d'I / O peut commencer, celle-ci est exécutée en parallèle avec la suspension du processus CPU.

f) lorsque l'entrée - sortie est terminée, le processus qui l'a réalisée meurt et celui qui l'avait demandée est réveillé.

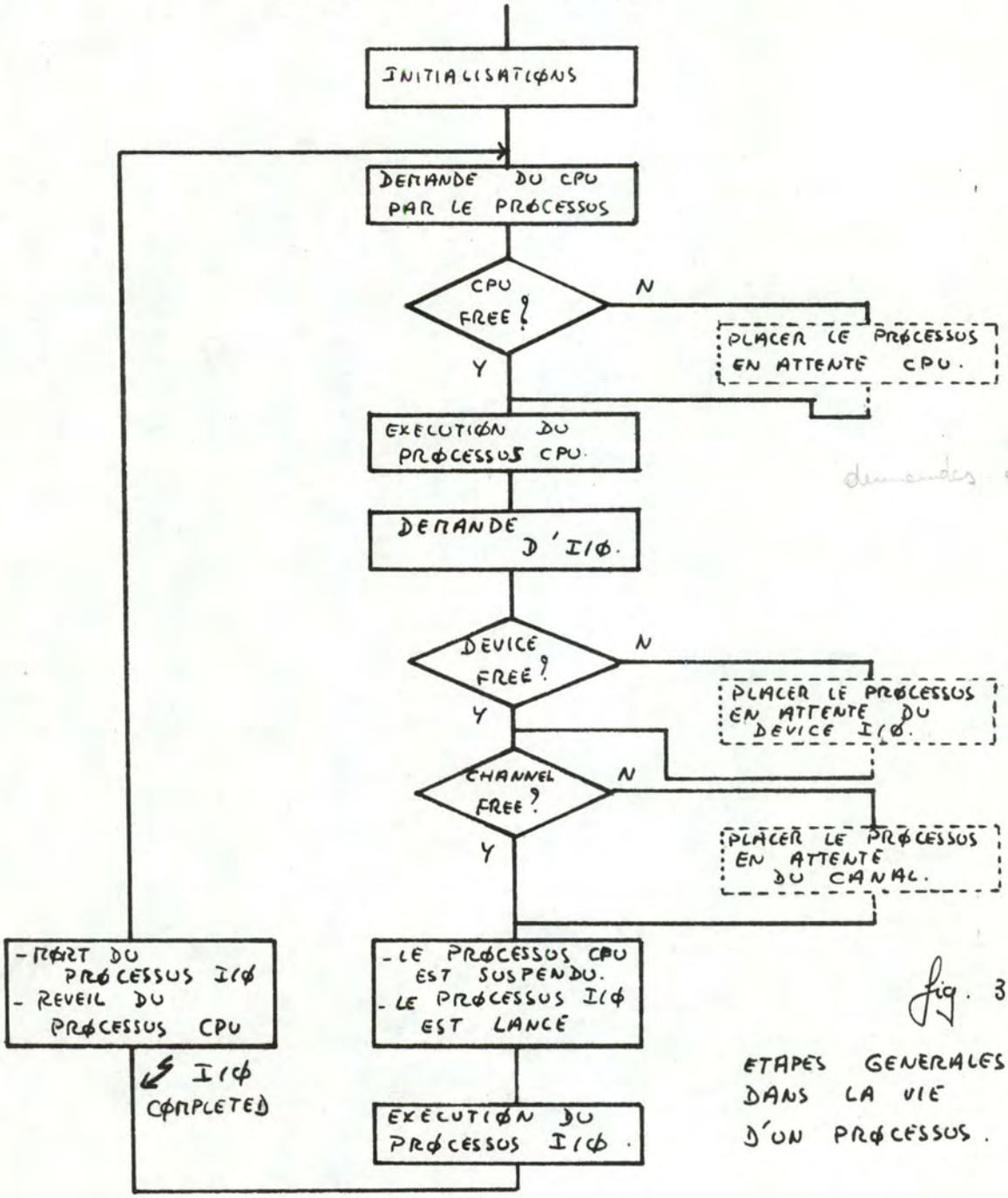


fig. 3.2.

ETAPES GENERALES DANS LA VIE D'UN PROCESSUS.

Il est bien entendu que cette approche reste très générale et qu'il serait nécessaire pour la compléter, de pénétrer plus avant dans le détail des mécanismes sous-jacents, particulièrement en ce qui concerne les entrées-sorties. Ce développement sera réalisé dans la suite, lors du sous-chapitre consacré à la gestion simulée des processus.

3.1.2. Optique simulation

Il est utile de rappeler l'optique dans laquelle l'évaluation des performances d'un operating system est envisagée dans le cadre de ce travail. L'objectif est de parvenir à élaborer un outil de simulation grâce auquel l'analyste de système soit en mesure de dégager des normes de comportement du produit étudié. Le critère adopté dans la construction du modèle est de représenter et de schématiser aussi fidèlement que possible les données et les procédures réelles, tout en restant en-deçà d'un niveau de simulation où le degré de précision apporté se révèlerait, dans une première phase tout au moins, dénué d'opportunité réelle.

Le chapitre II a examiné les problèmes résultant de la charge de travail soumise au système. Le début du chapitre III est consacré aux principes de base qui régissent la vie d'un processus, vue par l'O.S. ; la suite va s'attacher à préciser l'impact de ces notions dans l'optique choisie, de façon à intégrer celles-ci dans une gestion simulée des processus et des ressources qui soit réaliste.

- - - - -

3.2. GESTION SIMULEE DES PROCESSUS

3.2.1. Restriction

Avant d'envisager ce qui doit constituer le noyau de la simulation, il est nécessaire de considérer la gestion des processus au niveau des grands principes qu'elle applique. Il serait illusoire et de plus inutile de prétendre introduire dans le modèle tous les mécanismes intervenant dans une telle gestion. En d'autres termes, le concepteur ou l'analyste de système doit se fixer un niveau d'intérêt ou de détail sur base d'un ou de plusieurs critères bien définis et non contradictoires.

A titre exemplatif, ces critères peuvent être :

a) le temps : si le simulateur contient une horloge (Graham [3]) on décide de ne pas descendre plus bas que la microseconde, ou même la milliseconde.
b) le choix des fonctions à simuler : pour un processus CPU, on suppose, étant donné une certaine tranche de temps simulé, que la procédure d'adressage à réaliser lors de l'exécution de chaque instruction (dans le contexte d'une mémoire virtuelle, p. ex.), ou la détection et le traitement de certaines erreurs (overflow, division par zéro,...) ne constituent pas des facteurs qu'il est intéressant d'introduire dans un modèle d'évaluation globale des performances d'un O.S.

Il est bien entendu que ceci est à prendre en considération à titre informatif uniquement, étant donné que c'est finalement le responsable au niveau du système qui déterminera les points principaux (ainsi qu'éventuellement les points secondaires) sur lesquels l'étude d'estimation doit porter plus spécialement.

3.2.2. Processus simulé

Les éléments dont on dispose jusqu'à présent pour aborder la notion de processus simulé sont les suivants :

a) une schématisation de la charge de travail (chapitre II).

b) le concept d'événement, vu par le système.

La représentation du workload traduit la séquence d'appel des ressources par le processus, et ce sont les différents événements qui caractérisent la vie de celui-ci. En simulation toutefois, la signification que l'on accorde à un événement aura une portée plus grande que celle que lui donne le système. En effet, au lieu de considérer comme génératrices uniques d'un événement, les ruptures dans la vie d'un processus (c'est-à-dire les changements de son état), on procède à une extension de la notion et on se réserve la possibilité de distinguer plusieurs événements à l'intérieur même d'une tranche de temps simulé où le processus garde le même état. L'intérêt de ceci réside évidemment dans le fait de pouvoir distinguer plusieurs phases logiques dans un même intervalle de temps simulé au cours duquel le processus utilise une même ressource. Des applications de ce principe seront réalisées :

- lors d'une opération d'entrée - sortie simulée.
- lors de l'intégration dans le modèle des mécanismes simulés d'un algorithme de mémoire virtuelle segmentée.

Si l'on considère maintenant l'existence même du processus, on constate deux choses :

- d'une part, lors de l'exploitation effective du système, celui-ci gère un certain nombre de processus existant réellement.
- d'autre part, lorsqu'il est question de simuler le comportement du système, le contexte précédent

est bouleversé de la manière suivante : l'objet simulé (le système) n'existant plus que par l'intermédiaire de sa représentation au sein du modèle (objet simulant), les processus n'ont donc plus de réalité intrinsèque. L'objet simulé devient un ensemble de données que le modèle traite au moyen d'opérateurs de simulation.

Par conséquent, il est nécessaire de prendre les actions suivantes :

- en ce qui concerne le système évalué, sa représentation doit constituer le coeur même du modèle (1), en ce sens que les différentes décisions prises dans le réel doivent posséder ou non, selon leur degré d'importance, leur équivalent dans le modèle.
- chaque processus sera présent lors de la simulation par le biais de son descripteur, qui constitue un moyen assez facile de pallier à l'absence de l'entité réelle.

Le noyau du modèle étant examiné dans la suite, voyons maintenant l'intégration d'un processus dans la simulation. Le rôle du descripteur de processus est essentiellement de garder à jour l'état courant de l'entité qu'il représente. En effet, pour pouvoir opérer une gestion correcte, le modèle doit avoir la possibilité d'interroger à tout moment (excepté dans des conditions bien précises signalées ultérieurement) les différents éléments dont il a la responsabilité. Le descripteur de processus est l'un d'eux ; il peut être représenté par un vecteur (fig. 3.3.) et doit contenir en permanence les renseignements qui permettront au simulateur de connaître :

---

(1) Une des conditions de validité du modèle exige que les fonctions simulées soient identiques, au moins quant à leur fonctionnement global sinon quant à leur complexité, aux fonctions réelles.

a) dans le cas général :

- la priorité du processus (si du moins dans le système réel les processus sont affectés d'une priorité).
- la référence du prochain événement intervenant dans la vie du processus.
- la durée du temps simulé pendant laquelle le processus doit encore utiliser la ressource qu'il occupe actuellement (ou qu'il occupera dans la suite s'il est prêt à le faire) avant que l'événement mentionné ci-dessus ne survienne.
- une référence signalant le début des opérations à simuler ; le modèle doit la prendre en considération pour lancer le processus, ou le relancer si celui-ci est cyclique et qu'il a accompli la dernière phase de la boucle.
- une référence signalant l'endroit où le processus a été suspendu ou mis en attente ; c'est en effet à partir de là qu'il doit continuer à vivre.

b) dans le cas d'un processus d'entrée - sortie :

une opération I /O étant plus complexe qu'un accès en mémoire centrale, et nécessitant un temps d'exécution beaucoup plus grand, il est intéressant de descendre dans le détail de son fonctionnement ; cette approche sera réalisée lors du traitement simulé d'une entrée - sortie. En attendant, il est opportun de préciser la nature des différents renseignements dont le modèle aura besoin pour simuler dans de bonnes conditions l'opération demandée :

- le n° du cylindre (si un disque est concerné) à partir duquel on peut retrouver (ou mémoriser) l'information voulue.
- la référence du dispositif d'entrée - sortie.
- la longueur de l'enregistrement que l'on désire lire ou écrire.
- la référence du fichier concerné.

- la nature de l'opération sollicitée (lecture - écriture).

Il est à noter que l'introduction ci-dessus d'éléments dont la valeur serait une adresse interviendrait déjà à un niveau assez détaillé d'une gestion simulée de l'utilisation des ressources en capacité de mémorisation.

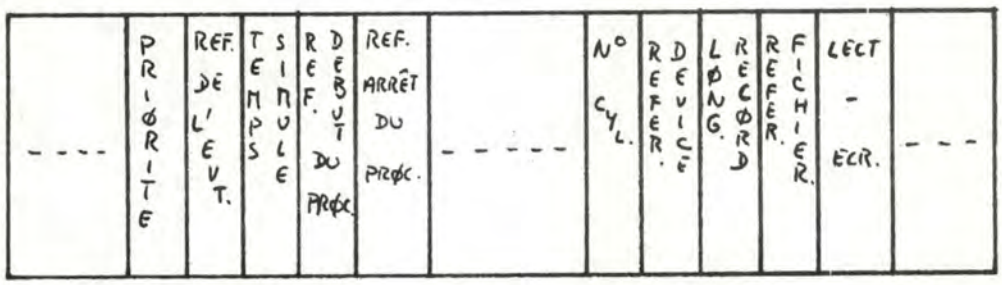


fig. 3.3.

Outre ces informations, il est nécessaire évidemment d'assurer au modèle la possibilité de retrouver le descripteur du processus d'une façon quelconque ; toutes les opérations de recherche d'informations doivent être prises en charge par le réalisateur de l'implémentation, si du moins celui-ci utilise un langage non spécialisé. Enfin, il est utile de préciser que dans le cas d'une complexification de la gestion des processus et des ressources (exemple : l'utilisation d'une technique de mémoire virtuelle), le descripteur pour lequel on a suggéré ci-dessus un contenu devra très probablement être complété.

3.2.3. "Process management" (1)

Après avoir examiné le problème de l'existence d'un processus simulé au sein du modèle, il faut envisager ceux qui vont se poser au cours de la vie, si on peut s'exprimer de cette manière, de ce processus.

---

(1) voir Mac Dougall [9]

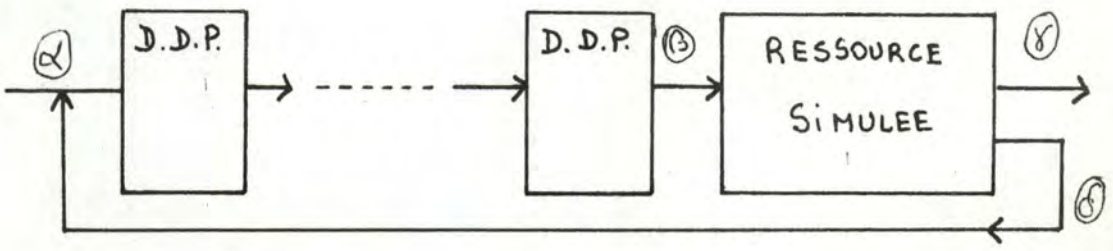
A) Le problème des files d'attente

Il va de soi qu'il n'est guère intéressant d'élaborer un outil d'évaluation de performances de système si la stratégie de gestion des processus ne permet que la monoprogrammation.

Par contre, s'il est possible d'utiliser la multiprogrammation, l'aspect concurrentiel des processus peut se manifester à chaque demande d'allocation d'une ressource quelconque. Il s'ensuit un problème de file d'attente auquel l'O.S. apporte une solution particulière. La gamme des algorithmes applicables à une telle situation est très large et il n'entre pas dans le but de ce travail d'examiner l'impact qu'ils pourraient avoir chacun au niveau du modèle de simulation. Certains d'entre eux cependant sont envisagés dans les lignes qui suivent.

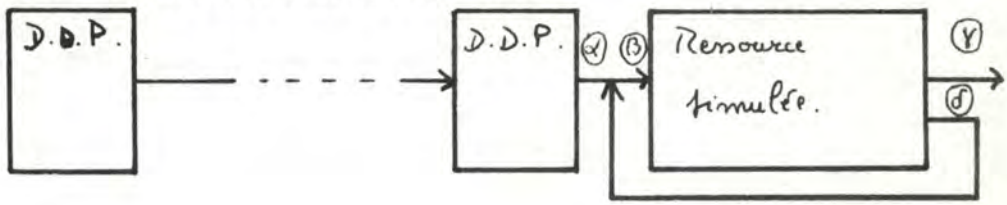
Soit une ressource partageable(l'unité centrale, un dispositif d'entrée - sortie, ...) sollicitée par différents processus. Au niveau de la simulation, et pour différentes techniques de gestion de ces processus, le problème peut être schématisé de la façon suivante :

a) Procédure FIRST IN - FIRST OUT



- la ressource simulée est représentée par son descripteur.
- D.D.P. : description de processus. Lorsqu'un processus (simulé) demande une ressource, son descripteur est placé au début de la file d'attente (ⓐ), l'occupation de la ressource ne pourra être simulée que lorsque le descripteur introduit en ⓐ atteindra le point ⓑ .
- lorsque la fin de l'utilisation simulée de la ressource intervient, le descripteur du processus quitte la file par le point ⓨ .
- la boucle ⓓ correspond à une légère complication de la procédure ; elle tend à attribuer à chaque processus un quantum de temps ; lorsque celui-ci est écoulé, le descripteur de processus quitte la file par le point ⓨ si l'exécution simulée est terminée ; sinon il est réintroduit au point ⓐ .

b) Procédure LAST IN - FIRST OUT

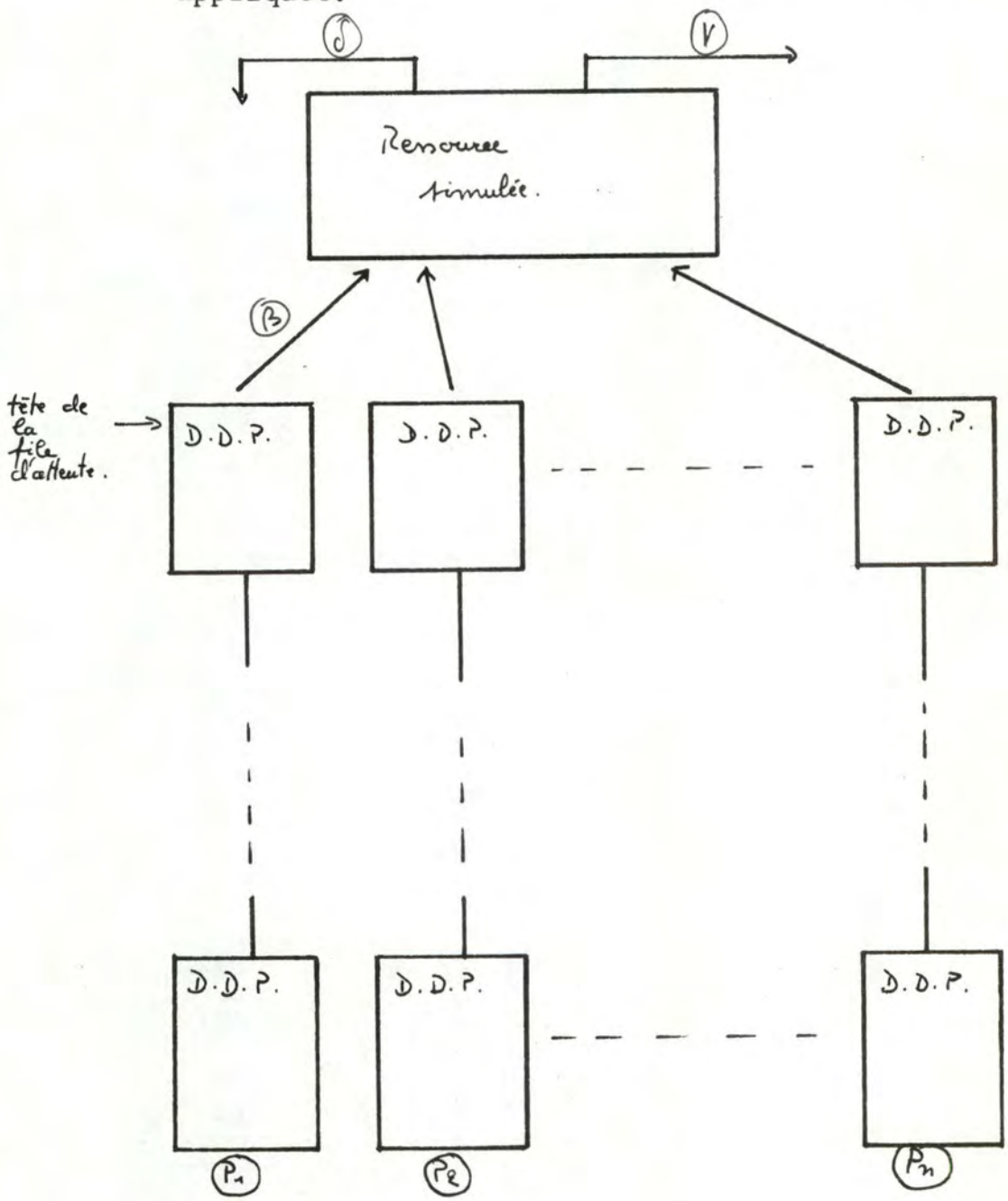


Etant donné les légères modifications apportées à la technique décrite au point a) , il n'est pas nécessaire de fournir de plus amples explications.

c) Utilisation de priorités (avec préemption)

A chaque processus on affecte une priorité d'exécution, d'occupation d'une ressource. Le cas typique est la concurrence de processus en attente du CPU. Il est nécessaire ici de sauvegarder la valeur de la priorité accordée, et la meilleure façon pour

le faire est sans doute, comme on l'a prévu auparavant, de réserver à cet usage un élément du descripteur de processus. En effet, à la différence d'un critère comme le quantum de temps, qui appartient plutôt à l'environnement du système lui-même, la notion de priorité, elle, apparaît comme caractéristique propre du processus auquel elle est appliquée.



- on prévoit une file d'attente par priorité P_i.
- la permission de l'accès à la ressource simulée ((β)) est accordée à l'élément en tête de la file d'attente qui vérifie les 2 conditions suivantes :
  - elle est non vide.
  - il n'existe pas de file d'attente non vide dont les éléments soient affectés d'une priorité supérieure à la sienne.
- lorsque le processus simulé "abandonne" la ressource simulée, ce peut être pour 2 raisons :
  - l'exécution simulée est réellement terminée. ((γ))
  - ou le processus simulé a dû céder la place à un autre (à cause du phénomène de préemption). Dans ce cas, le descripteur du premier cité réintègre la file d'attente voulue ((δ)). Le choix de la procédure appliquée pour effectuer cette rentrée n'est pas toujours fixé de manière immuable, et peut, selon le cas, faire appel par exemple à la technique FIFO ou LIFO.

De façon globale, le mécanisme de file d'attente pour les descripteurs de processus est utilisé chaque fois qu'une ressource simulée quelconque est susceptible d'être "partagée" (1) entre plusieurs processus simulés. D'où l'importance réelle à donner au terme de ressource. Les cas d'utilisation de file d'attente que l'on peut considérer comme relativement évidents concernent les dispositifs suivants :

- l'organe central de traitement. On touche en fait ici à un des problèmes majeurs à résoudre lorsqu'on veut doter le système et la configuration de performances respectables : comment assurer la

(1) le partage est à considérer comme un multiplexage, ou partage selon le mode exclusif.

disponibilité optimale du CPU aux utilisateurs, et comment dès lors effectuer dans cette optique la gestion des processus ?

- les dispositifs d'entrée - sortie ; on distingue :
  - les organes de mémorisation eux-mêmes (avec l'unité de contrôle).
  - les canaux de transfert.

  On peut affirmer qu'à ce niveau-ci se situe le second volet du problème soulevé ci-dessus : en effet, en même temps que se pose la question de la meilleure utilisation du CPU, existe la difficulté de réserver aux unités d'entrée - sortie une occupation optimale.

  Note : Il est certain d'une part que le fait de parler d'optimalité n'engage à rien en lui-même, et d'autre part que la phase la plus intéressante dans la recherche d'un tel but est constituée par la quantification et la réalisation des objectifs. Mais c'est là un domaine nouveau à l'intérieur duquel il n'est pas prévu de s'introduire, tout au moins dans le cadre de ce travail.

Si l'on considère les ressources évoquées ci-dessus, on constate immédiatement que leur nature est essentiellement hardware. Ce serait pourtant une erreur de se limiter à cette conception des choses. En effet, il existe un exemple de ressources au niveau du software que l'on doit gérer selon les mêmes principes, en ce sens que la propension de processus simultanés à tenter de s'assurer en parallèle du contrôle et de l'utilisation de certaines séquences de code doit avoir systématiquement un effet nul.

L'emploi de files d'attente est un moyen commode pour assurer une telle gestion dans le système réel, et il n'existe aucune raison pour ne pas adopter en simulation la même philosophie vis-à-vis des descripteurs de pro-

cessus. Au niveau de l'implémentation, ceci peut se traduire par exemple par une utilisation de verrous ou de sémaphores.

B) Le process scheduling (1)

On peut affirmer qu'avec cette partie on atteint véritablement le coeur du problème. En effet, pour le système réel, il s'agit d'assurer aux différents processus concurrents la poursuite de leur vie dans les meilleures conditions. A cette fin, l'O.S. dispose d'un algorithme de scheduling conçu de telle manière qu'il puisse, en principe, satisfaire au mieux les différents besoins en ressources exprimés.

Au niveau de la simulation, il est de toute évidence très important d'intégrer fidèlement dans le modèle les mécanismes de scheduling tels qu'ils apparaissent à l'intérieur du système. Les entités de base sur lesquelles on va jouer sont constituées par les descripteurs de processus, et le processeur à propos duquel on simule un scheduling n'est autre que le CPU.

Pour un processus simulé déterminé, on distingue généralement deux types d'événements différents :

a) d'une part ceux qu'on pourrait appeler "internes" au processus simulé : il faut entendre par là tous les éléments dont l'occurrence peut être engendrée par le biais du processus simulé lui-même (2) ; le délai ΔT de cette occurrence est relatif, étant donné l'existence de plusieurs processus

---

(1) Il est nécessaire en fait que la simulation rende compte du dispatching des processus, et dans une moindre mesure, sauf si l'intérêt principal se situé à ce niveau, du scheduling étant donné que ce terme s'applique plus généralement à des fonctions moins profondes du système.

(2) C'est en fait la procédure de contrôle du processus simulé qui se charge de la génération.

simulés compétitifs vis-à-vis de la même ressource dans un même laps de temps.

b) d'autre part les événements "externes" au processus simulé. Il s'agit là d'événements qui sont attendus par le processus simulé, tout en restant indépendants vis-à-vis de lui quant à la détermination de leur moment d'occurrence. Ils traduisent en fait des signaux externes destinés au processus simulé; selon le cas, l'utilisation de ces signaux peut être différée, ou au contraire doit être réalisée dès qu'ils arrivent.

De façon un peu plus formelle, en supposant l'utilisation d'une horloge de temps simulé, le modèle doit considérer les éléments suivants, pour un nombre quelconque de processus simulés :

- soit T la valeur du temps simulé à un certain moment.
- dans un environnement de multiprogrammation, plusieurs processus simulés se partagent le CPU ; on aura donc, pour chacun d'eux, une valeur ΔT traduisant la tranche de temps simulé dont ils ont encore besoin pour exécuter un travail donné sur le CPU.
- il se peut enfin que, parallèlement à tout ceci, un ou plusieurs simulés attendent, pour continuer à vivre, un signal externe dont l'occurrence est fixée au temps simulé T', T",... (chacun étant $>$ T) selon le processus concerné.

Selon la stratégie adoptée, le scheduling simulé donnera les résultats suivants :

a) <u>dans le cas FIFO</u> :
   - on considère le ΔT contenu dans le descripteur de processus ($\overline{11}$) simulé candidat immédiat à la sortie de la file.
   - on considère parmi les valeurs T', T",... celle qui est la plus proche de la valeur T ; admettons que ce soit T*.

- on compare $T + \Delta T$ avec T^* :
  *) si $T + \Delta T > T^*$: l'événement qui sera généré en premier lieu par le simulateur sera celui qui doit arriver en T^* ; le processus simulé concerné par cet événement sera averti de l'occurrence de celui-ci (1).

De plus, étant donné que le parallélisme des processus simulés joue, on doit soustraire à ΔT la quantité de temps$(T^* - T)$, pour connaître le temps simulé $\Delta T - (T^* - T)$ dont a encore besoin le processus simulé (π) en CPU. Pour la suite, on recommence la procédure depuis son début, après avoir mis T à jour en faisant $T = T^*$, et enlevé T^* de la liste T', T",...

*) si $T + \Delta T < T^*$, l'événement généré en premier lieu sera celui qui doit arriver en $T + \Delta T$.
L'horloge est mise à jour par $T = T + \Delta T$.
Le descripteur du processus simulé contenant le ΔT sort de la file d'attente et on continue le scheduling.

*) si $T + \Delta T = T^*$: l'ordre de génération des événements n'a plus alors aucune importance, puisque dans la réalité ils sont sensés arriver en même temps, ce qui est certainement rarissime.

b) <u>dans le cas des priorités</u> :
- on considère la lère file d'attente non vide qui possède la plus forte priorité.
- on considère le ΔT contenu dans le descripteur de processus (π 1) simulé se trouvant en tête de la file.
- on considère la plus petite valeur T^* parmi les éléments de la suite T', T", ...
- on compare $T + \Delta T$ avec T^*.

---

(1) son descripteur est alors introduit dans la file d'attente FIFO. Ceci suppose, si l'introduction est réalisée de manière conforme à la règle, que l'utilisation du signal peut être différée. Ce n'est pas toujours le cas.

*) si $T + \Delta T > T^*$: l'événement généré en 1er lieu sera celui dont le moment d'occurrence est T^*, le processus simulé concerné par cet événement sera averti de l'arrivée de celui-ci et une action sera prise en conséquence (soit on différe l'utilisation du signal, soit on affecte au processus la plus haute priorité combinée avec un ΔT nouveau suffisamment petit que pour assurer à ce processus simulé la garantie d'être traité de suite).
Il faut de nouveau mettre à jour le ΔT associé au processus simulé (Π 1) en effectuant $\Delta T = \Delta T - (T^* - T)$. On recommence alors la procédure à son début, sans avoir oublié de mettre à jour T par l'opération $T = T^*$, et d'enlever T^* de la suite T', T'', ...

*) si $T + \Delta T < T^*$: l'événement généré en 1er lieu sera celui qui doit arriver en $T + \Delta T$.
Le modèle met l'horloge à jour en effectuant $T = T + \Delta T$. Le descripteur du processus simulé concerné sort de la file d'attente, et on continue le scheduling.

*) si $T + \Delta T = T^*$: même raisonnement que pour FIFO.

On vient d'examiner, pour les stratégies FIFO et scheduling par priorité, le problème en simulation de la sortie d'un descripteur de processus simulé hors d'une file d'attente constituée devant la ressource CPU. Le problème de l'entrée dans une file peut se résoudre assez rapidement de la manière suivante :

- le ΔT relatif à un événement donné pour un processus donné est évalué.
- on place ensuite les valeurs (n° événement , intervalle de temps ΔT) dans le descripteur concerné, que l'on introduit à sa place dans la file adéquate, selon le type de stratégie adopté.

Les mécanismes décrits ci-dessus tels que le modèle de simulation les considère doivent assurer le progrès dans le temps des processus simulés par la génération, pour chacun d'eux, du prochain événement qui les caractérise. Ceci suppose que le modèle ait la possibilité de connaître l'événement dont il doit simuler l'occurrence. Sur ce point, une schématisation correcte du workload doit constituer une aide efficace à la réalisation de l'objectif fixé.
Chaque système possédant ses propres particularités sur le plan du scheduling des processus, il est évident que les cas envisagés ci-dessus ne constituent que des exemples de stratégies éventuelles et que les caractéristiques intrinsèques de l'algorithme simulé doivent figurer le plus fidèlement possible à l'intérieur du modèle.

C) Initialisation et mort du processus simulé.

C'est un problème dont les paramètres dépendent fortement du responsable de l'évaluation du système. En effet, en vertu de différentes raisons, il peut par exemple :

- souhaiter un nombre donné de processus simulés concurrents.
- souhaiter que des "workloads" de types donnés soient introduits simultanément (ou en séquence, ou selon différentes combinaisons) sous forme de leur schématisation dans la simulation.
- souhaiter que les charges de travail simulées soient prises en compte un certain nombre de fois.

En conséquence, le responsable devra formuler les conditions dans lesquelles il désire voir réaliser la simulation. Quelles que soient ces conditions, le modèle devra résoudre le problème de l'initialisation et de la mort du processus simulé. A ce sujet, on peut remarquer le fait suivant : la localisation dans le

temps et la nature de ces procédures doivent être considéréesgénéralement comme étant indépendantes de tous les processus simulés eux-mêmes ; il existe toutefois des exceptions où la naissance d'un processus simulé peut être provoquée par une action prise à l'intérieur d'un autre processus. En vertu de ces considérations et en fonction des types d'événement relevés précédemment (cf. supra p. 67 , 68), on peut envisager le début et la fin d'un processus simulé comme étant associés à l'occurrence d'un événement externe au processus, étant entendu que le signal provoqué par cet événement doit être utilisé immédiatement de façon à réaliser :

- soit le lancement du processus simulé.
- soit la destruction du processus simulé.

Il est à noter que cette dernière opération ne doit pas impliquer nécessairement la destruction du descripteur de processus ; elle peut se résumer à écarter celui-ci des files d'attente correspondant aux différentes ressources et à le placer dans une liste reprenant par ex. tous les descripteurs de processus dont le temps à simuler est arrivé à terme.

D) Le problème des interruptions

Etant donné les nombreux cas d'interruption prévus lors de la conception d'un système, les possibilités d'intégrer un tel mécanisme dans un modèle de simulation sont assez variés. Néanmoins, il ne faut pas oublier le critère important d'opportunité auquel la décision d'introduire ou de négliger telle ou telle fonction de l'operating system dans le modèle d'évaluation doit se référer. Une application de ce critère se situe certainement au niveau du choix des interruptions que l'on décide de simuler ou non.
En ce qui nous concerne, les cas d'interruptions retenus dans ce travail sont uniquement ceux qui sont

relatifs au traitement des entrées - sorties (1), pour lequel on suppose de plus un déroulement sans incident.

La simulation d'une entrée - sortie est confrontée à deux problèmes : quand et comment doit-on la réaliser ?

1) Quand simuler une I/O ?

La solution à cette question dépend directement de la schématisation du workload et de l'interprétation de celle-ci. En effet, comme on l'a vu au chapitre II (p. 40), l'input du simulateur doit reprendre, formulées de façon quelconque mais fidèle à l'aspect structurel et quantitatif des applications, les demandes simulées de ressources disponibles. Les dispositifs d'entrée - sortie font partie de ces ressources, et il est dès lors normal de prévoir dans la représentation d'un traitement y faisant appel, les opérations adéquates.
Le modèle d'évaluation doit en conséquence, lorsqu'il rencontre dans l'input ce qui correspond à une demande d'entrée - sortie, simuler le fonctionnement sous-jacent à celle-ci.

2) Comment simuler une I/O ?

La complexité des procédures d'entrée - sortie rend nécessaire, peut être plus qu'ailleurs, une approche synthétique des opérations réelles. En effet, il serait inutile, à moins que ce soit là l'objectif fixé, de simuler de façon trop détaillée le déroulement d'une opération I/O à l'intérieur d'un modèle d'évaluation qui veut avant tout dégager de façon globale des normes de comportement du système. En conséquence, il semble donc intéressant d'aborder les entrées - sorties en considérant les phases principales qui interviennent dans leur réalisation.

---

(1) Seuls les I/O avec transfert de données sont considérés.

Sur cette base, on peut établir le schéma logique suivant (fig. 3.4) qui présente une découpe assez fonctionnelle d'une opération I/O complète.

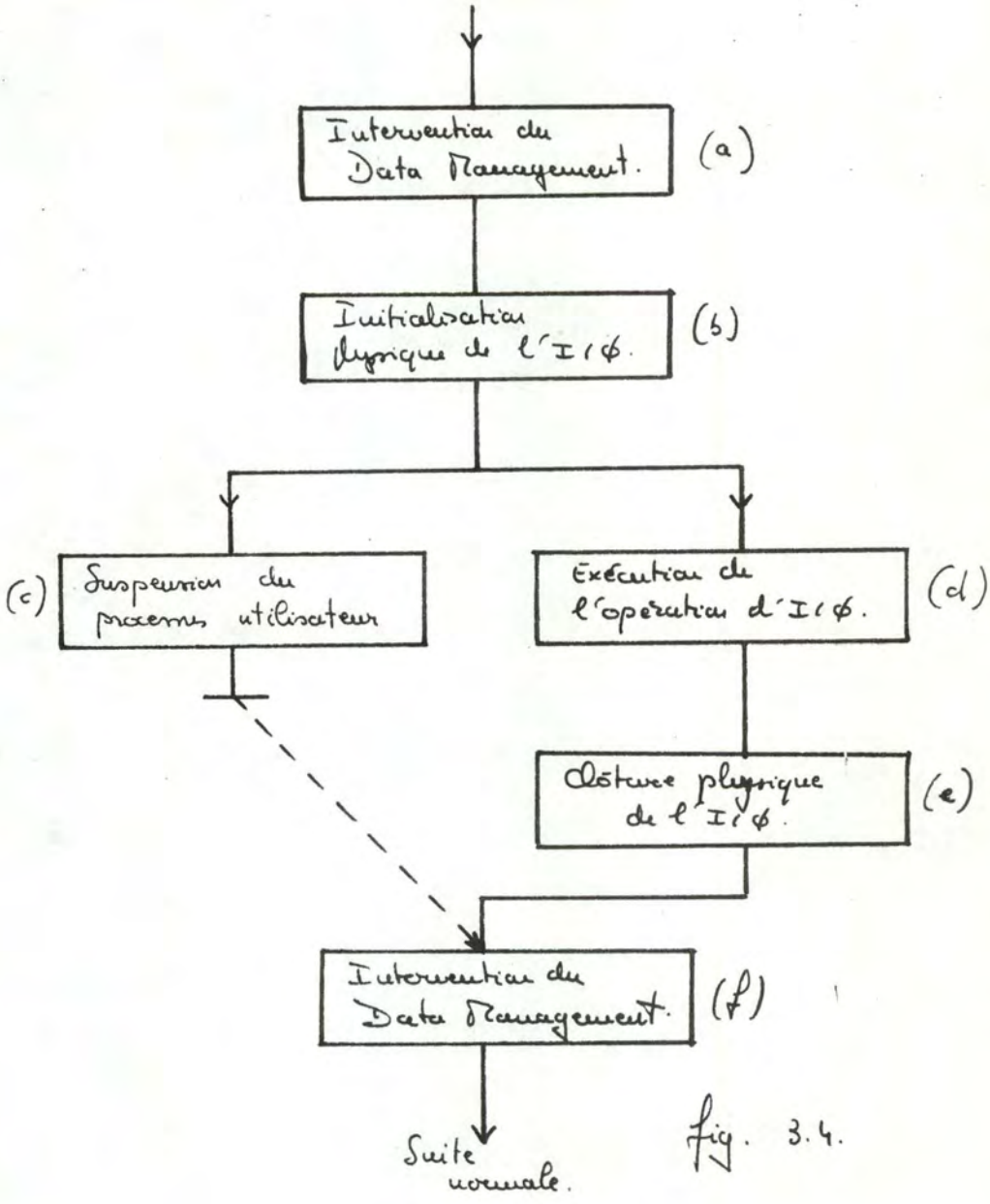


fig. 3.4.

Ce schéma permet en outre d'observer le fonctionnement général du système d'entrée - sortie sous l'angle particulier qui nous intéresse, à savoir la gestion des processus et la gestion des ressources.

En effet, si l'on reprend les différents blocs du diagramme, on peut constater ceci :

a) l'intervention du Data Management : elle sert à régler des problèmes tels que la validation de l'ordre programmé, la méthode d'accès spécifiée, la gestion des buffers, le blocage, le déblocage, etc... On considère que l'exécution de ces fonctions diverses fait partie intégrante du processus - utilisateur, la ressource dont celui-ci fait usage étant le CPU.

b) l'initialisation physique de l'I/O : est réalisée par un ensemble de routines du système (incluant S I O, ...) dont la disponibilité est assurée à tous les utilisateurs : de nouveau, cette exécution est considérée comme partie intégrante du processus - utilisateur, la ressource concernée étant toujours le CPU.

c) la suspension du processus - utilisateur : est associée à l'ordre d'attente qui suit normalement la demande d'une entrée - sortie. Cette suspension a pour conséquence immédiate la perte du CPU par le processus - utilisateur.

d) l'exécution de l'opération I/O, effectuée en parallèle avec l'opération de suspension (point c) comprend la recherche et le transfert des données.
La réalisation de ces fonctions est assurée par un processus que l'on peut qualifier de hardware, et dont la durée de vie est limitée à l'exécution de ces seules opérations.

e) la clôture physique de l'I/O est assurée par un ensemble de routines du système, non distribuées aux utilisateurs. Le processus sous-jacent est un processus - système et la ressource concernée est le CPU.

f) la clôture logique de l'entrée - sortie est réalisée par le processus - utilisateur que l'on vient de réveiller. Avec cette opération se termine la séquence des différentes phases intervenant dans la réalisation d'une I /O.

Etant donné la netteté de la découpe et celle de l'attribution des différentes fonctions aux processus adéquats, il est possible d'intégrer facilement dans le modèle d'évaluation les mécanismes aptes à réaliser la simulation d'une entrée - sortie ; l'élément de base sur lequel celle-ci s'appuie reste le descripteur de processus, dont le rôle consiste essentiellement à tenir à jour l'état de l'entrée qu'il représente.

Chacun des processus mis en cause requiert, afin de mener à bien les fonctions diverses dont il est chargé, une certaine durée de vie, décomposée en une ou plusieurs tranches, qu'il est nécessaire de prendre en compte au niveau de la simulation.

Sans vouloir ici respecter l'exactitude des proportions, un diagramme des temps simulés peut s'établir de la façon suivante (fig. 3.5), plaçant certaines opérations en correspondance évidente, et faisant ainsi apparaître une hiérarchisation à l'intérieur du déroulement d'une entrée-sortie.

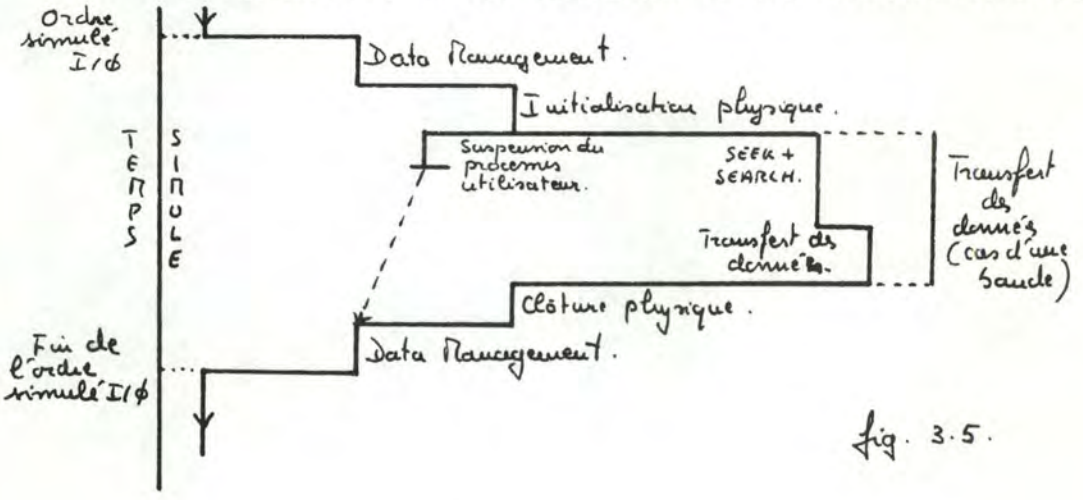


fig. 3.5.

Après avoir, selon le point de vue de la simulation, examiné sous l'angle fonctionnel d'abord, sous l'angle type de processus ensuite, le déroulement global d'une entrée - sortie, il est nécessaire de se pencher, toujours dans la même optique, sur certains problèmes plus précis dont on peut d'ailleurs deviner l'existence au vu de ce diagramme :

- l'examen de la disponibilité effective des ressources pour les processus voulus.
- la suspension et le réveil du processus - utilisateur.

a) La disponibilité des ressources simulées

On peut distinguer à ce sujet deux classes de ressources :

1/ Le CPU : tout processus simulé demandant le CPU est soumis aux règles de scheduling telles qu'elles sont implémentées dans le modèle d'évaluation. A ce propos, des exemples ont été donné lors d'un paragraphe antérieur (cf. supra p. 68).

2/ Les dispositifs d'entrée - sortie (1) : Le système réel met en oeuvre pour eux également une stratégie d'allocation qu'il est nécessaire de reprendre au niveau de la simulation. Quelle que soit cette stratégie, le modèle doit prévoir une représentation des dispositifs I /O telle qu'à chaque instant appartenant au temps simulé, il lui soit possible d'examiner si les conditions voulues sont remplies pour qu'un processus simulé puisse disposer des ressources qu'il

(1) De façon globale, le problème des entrées-sorties est lié également au problème du scheduling de haut niveau ; ce n'est pas cet aspect des choses qui est envisagé ici. Dans le cadre de ce paragraphe, on considère uniquement un aspect restreint, limité aux I /O que pourrait demander un processus - utilisateur.

requiert. En d'autres termes, on peut envisager par exemple d'intégrer dans un descripteur de chaque ressource (auquel le modèle peut accéder à tout moment) un indicateur dont le rôle consiste essentiellement à signaler au "dispatcher" des processus simulés, la disponibilité ou l'indisponibilité de la ressource. Dans ce dernier cas, le demandeur doit être placé en attente ; ceci peut se réaliser facilement en insérant le descripteur du processus dans la liste adéquate, qu'il sera nécessaire de consulter lorsque le processus simulé qui occupe la ressource à ce moment aura libéré celle-ci. Le cas typique mettant en oeuvre cette procédure, qui constitue en fait une application de la technique des verrous, est celui d'une entrée - sortie sur bande ou sur disque par exemple, dont la simulation est en partie différée, en raison de l'occupation d'une ressource requise par un autre processus à ce moment. Lorsque cette ressource sera libérée, l'examen de la file d'attente qui lui est associée doit permettre, si du moins il y a lieu de le faire, de continuer la simulation d'une opération I / O suspendue antérieurement. Les règles d'introduction et de suppression à l'intérieur d'une file d'attente déterminent quelle opération doit être prise en compte.

b) La suspension et le réveil d'un processus - utilisateur

Après avoir, dans le cadre d'une entrée-sortie, examiné une façon de traiter la suspension d'un processus simulé opérée pour cause d'indisponibilité de ressources, envisageons maintenant,

toujours dans un contexte identique, le même problème, mais dont l'existence est provoquée pour une cause différente de la précédente : lorsqu'il a lancé l'entrée-sortie, le processus - utilisateur est placé dans l'état "suspendu" (cf. supra, p. 53). En réalité, il doit attendre de pouvoir disposer des informations qu'il a demandées en mémoire centrale (si l'ordre était READ) ou de pouvoir disposer d'un buffer de sortie p. ex. (si l'ordre était WRITE) pour être en mesure de poursuivre sa vie dans l'état "actif" en ayant transité auparavant par l'état "prêt". La simulation doit prendre en compte cette difficulté et y apporter une solution qui, de nouveau en raison de l'optique choisie, permette au résultat de représenter assez fidèlement le déroulement d'une charge de travail donnée soumise au contrôle du système.

La résolution de ce problème suppose l'existence d'une réponse qui satisfasse à la fois les deux aspects sous lesquels celui-ci est posé :

a) il est nécessaire de disposer d'un mécanisme capable d'"endormir" le processus simulé lorsque celui-ci a lancé l'I/O simulée ;
b) il est tout aussi nécessaire de prévoir un mécanisme qui permette de "réveiller" le processus simulé "endormi" auparavant; cette opération doit être réalisée lorsque l'entrée-sortie simulée sera terminée.

A cet effet, les systèmes d'exploitation disposent d'un outil assez commode, appelé "sémaphore". Il est possible d'intégrer dans le modèle de simulation une fonction semblable

qui réalise automatiquement les opérations mentionnées ci-dessus. Dans le cas envisagé ici, on considère uniquement la suspension d'un processus simulé qui est due à une demande d'accès à un fichier, simulé également. On peut donc parler, de façon peu élégante peut-être, mais assez explicite, de "processus simulé X bloqué sur un accès ou fichier simulé Y"(1). La cause du blocage se situant au niveau des fichiers, on peut associer à chacun d'eux :

- une variable (qui fait office de compteur)
- une file d'attente destinée à contenir la liste des descripteurs de processus, lorsque les entités qu'ils représentent sont sujettes à une suspension simulée.

La combinaison et l'utilisation de ces éléments doivent permettre de gérer correctement les processus simulés lorsque ceux-ci provoquent une demande d'entrée-sortie (2).

En effet, on peut établir les procédures qui traduisent les diagrammes logiques suivants (fig. 3.6. et fig. 3.7).

1) Pour "endormir" un processus simulé, ou du moins tenter de le faire :

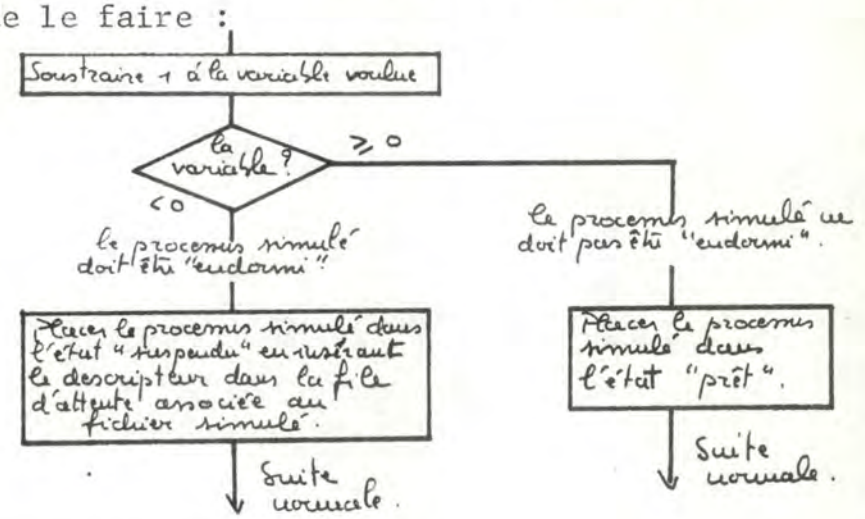


fig. 3.6

(1) Le mécanisme de suspension tel qu'il est employé ici peut soulever un problème de protection de fichiers assez complexe qui n'est pas abordé dans le cadre de ce travail.

(2) On suppose que l'ordre de lecture ou d'écriture est chaque fois immédiatement suivi de l'ordre d'attente de fin d'I/O correspondant.

2) pour "réveiller" un processus simulé s'il a été "endormi" :

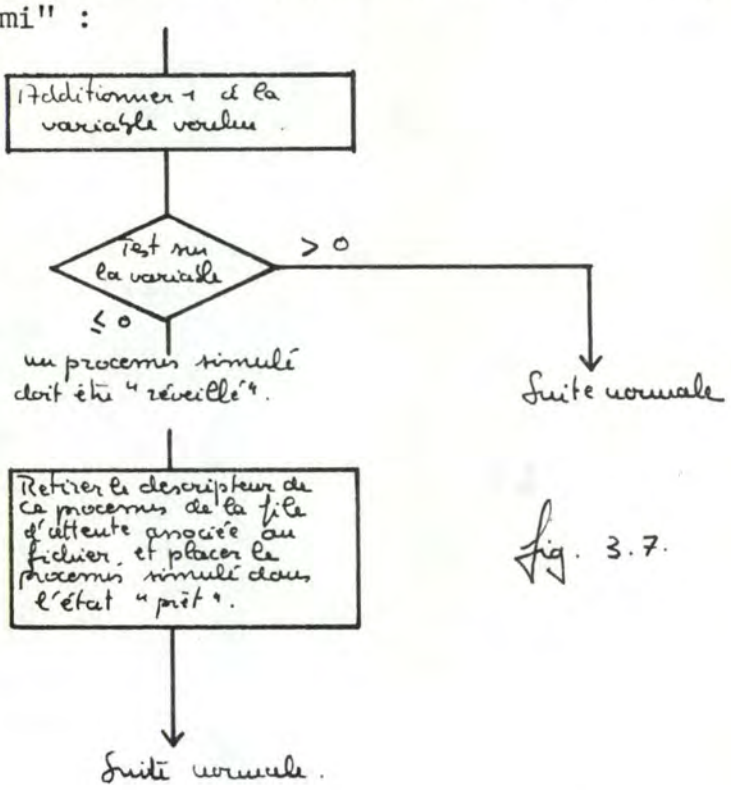


fig. 3.7.

La fig. 3.6 correspond à l'opération "P" effectuée sur un sémaphore. Elle représente le mécanisme à implémenter pour suspendre un processus simulé s'il y a lieu de le faire. En effet, il est possible que ce processus simulé puisse continuer à "utiliser" la ressource simulée CPU sans devoir nécessairement attendre la fin de l'entrée-sortie qu'il a lancée. Dans ce cas, la synchronisation demandée est inefficace et inutile. Par contre, si le test sur la variable révèle une valeur négative, le processus simulé pour lequel cette procédure est exécutée doit être "endormi" : ceci se réalise en enlevant le descripteur de la liste des descripteurs de processus "prêts" à utiliser la ressource CPU, et en le plaçant dans la file d'attente adéquate.

La fig. 3.7 correspond à l'opération "V" effectuée sur un sémaphore. Elle représente le mécanisme à implémenter pour réveiller un processus simulé sous

réserve que celui-ci ait été endormi auparavant. En effet, dans le cas, peu probable d'ailleurs, où la prise en considération et le traitement conséquent par le système de l'ordre d'attente qui suit généralement l'ordre d'entrée-sortie auraient une durée supérieure à celle de l'exécution de l'I/O, le processus - utilisateur n'a jamais été suspendu. Il est donc superflu, au niveau de la simulation, de vouloir replacer le descripteur de ce processus dans une file d'attente qu'il n'a jamais quittée. Dans le cas le plus fréquent, au contraire, cette opération est nécessaire étant donné que le descripteur du processus a été écarté de la file d'attente relative à la ressource simulée CPU.

E) Schéma global

Avant de terminer ce qui se rapporte au "process management" tel qu'il peut être considéré dans un modèle de simulation, il est utile de synthétiser les notions dont il a été question jusqu'ici et élaborer dans ce sens un schéma global qui livre ce qu'on pourrait appeler une photographie des entités sous contrôle du système.
Celui-ci considère un ensemble de processus qu'il doit gérer et un ensemble de ressources qu'il doit allouer, le tout suivant certaines règles bien définies. La simulation considère que pour elle, toutes ces entités n'existent pas et qu'en fait elles sont simplement représentées par un moyen quelconque. C'est sur cette base qu'elle travaille. A un moment donné du temps simulé, on peut supposer qu'il existe un certain nombre de processus simulés dans chacun des trois états possibles (actif - suspendu - prêt) et qu'ils occupent (ou sont en attente d'occuper) les ressources simulées voulues.

La fig. 3.8 donne un schéma général d'une telle situation ; on suppose que les processus sont affectés chacun d'une priorité d'accès au CPU.

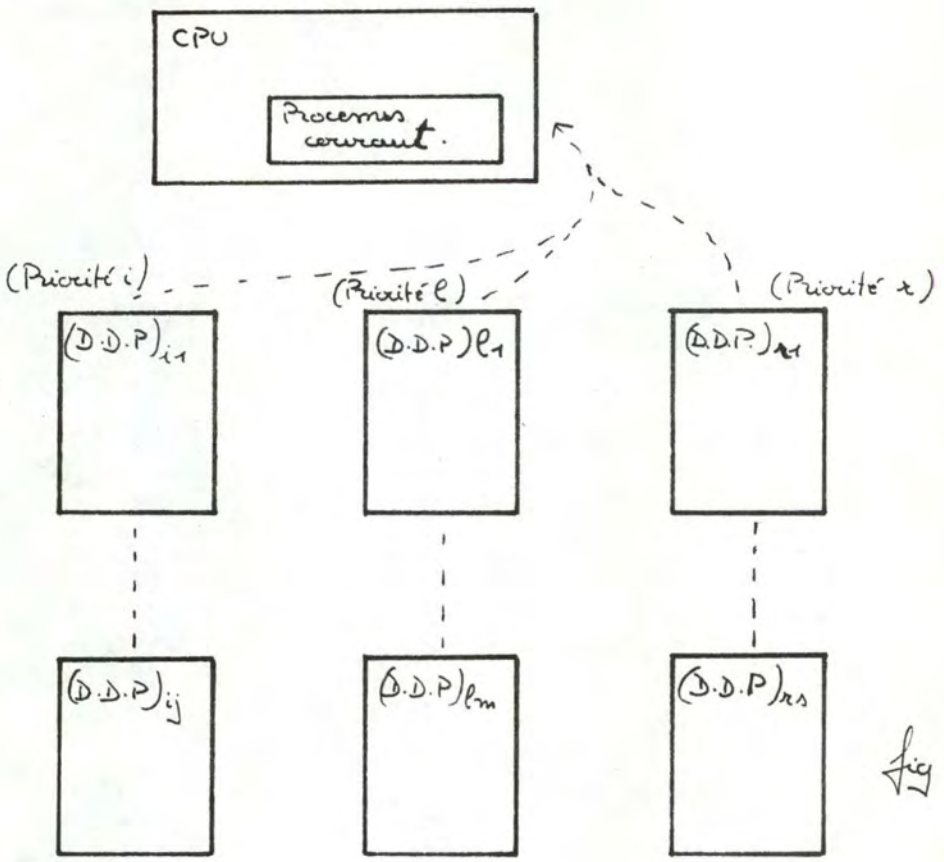


fig. 3.8 [a].

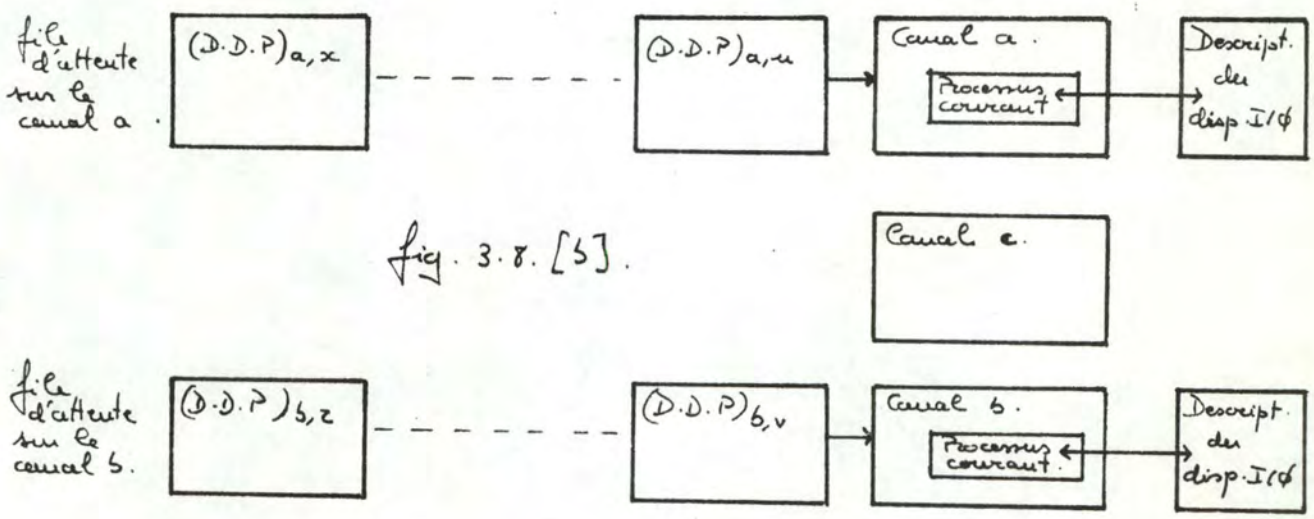


fig. 3.8. [b].

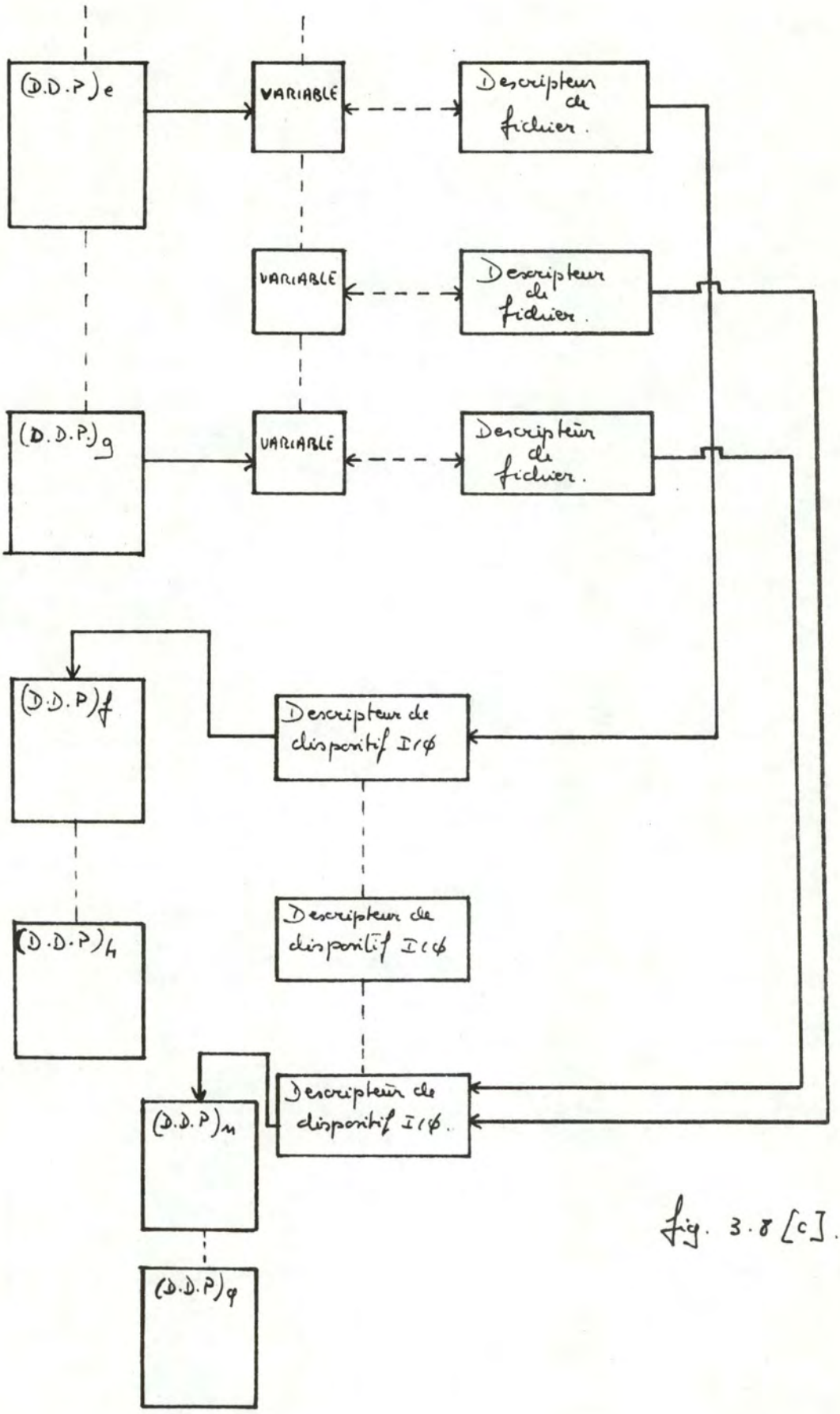

(D.D.P)e
VARIABLE
Descripteur de fichier.
VARIABLE
Descripteur de fichier.
(D.D.P.)g
VARIABLE
Descripteur de fichier.
(D.D.P)f
Descripteur de dispositif I/ϕ
Descripteur de dispositif I/ϕ
(D.D.P)h
Descripteur de dispositif I/ϕ.
(D.D.P)m
(D.D.P)q

fig. 3.8 [c].

La fig. 3.8 (a) reprend autour de la ressource CPU les processus simulés (représentés par le moyen de leur descripteur) qui entrent en concurrence pour la possession de celle-ci. On les divise en deux classes :

1) le processus simulé courant, qui dispose de la ressource simulée au moment T considéré.
2) les processus simulés dont l'état est "prêt". Ils sont répartis dans une série de files d'attente selon leur priorité.

La fig. 3.8 (b) est relative aux files d'attente qui peuvent se constituer devant chaque canal d'entrée-sortie. Pour chacun d'eux, il peut y avoir :

1) un processus simulé courant (hardware) "exécutant" l'entrée-sortie simulée mais aucune file d'attente.
2) aucun processus courant.
3) un processus simulé courant et une file d'attente. Ici, de même d'ailleurs que dans le cas 1), l'existence d'un processus simulé hardware courant signifie qu'il doit exister une liaison quelconque avec le descripteur du dispositif I/O concerné par l'opération en cours.

La fig. 3.8 (c) traduit la situation de processus simulés de nouveau liés à des opérations d'entrée-sortie.

1) les processus dont les descripteurs sont affectés d'un indice compris entre e et g ont demandé une entrée-sortie et son "bloqués sur un accès à un fichier" après examen de la variable dont il a été fait mention auparavant.
2) chacune de ces variables est associée à un fichier qui existe lui aussi en simulation par l'intermédiaire de son descripteur.
3) étant donné que l'on considère des fichiers supposés implémentés sur des dispositifs de mémorisation secondaires, il est nécessaire de prévoir entre les descripteurs des premiers et ceux des seconds les

liaisons voulues. Il doit évidemment être possible d'introduire la notion de dispositif multi-fichier.

4) au niveau de chacun de ces dispositifs, une liste peut se créer, comprenant un ensemble de processus simulés hardware qui sont en attente de pouvoir disposer du dispositif, et se trouver en mesure d'exécuter leur travail, si du moins le canal requis est libre. Le processus simulé hardware pour lequel ces conditions sont remplies devient alors courant pour ce canal.

3.2.4. Gestion des événements (1)

Nous avons abordé précédemment (cf. supra p. 53) la notion d'événement telle qu'elle est considérée au niveau du système. Il est possible d'introduire un tel concept dans un modèle d'évaluation de performances et de réaliser, dans le cas qui nous occupe, une "simulation par événements". On est toutefois amené à envisager l'occurrence d'un événement comme ne survenant pas uniquement chaque fois que se produit un changement d'état d'un processus simulé sous contrôle du modèle. En effet, il est plus intéressant d'étendre la notion et de donner à l'événement une signification plus large ; on considère au niveau de la simulation qu'à chaque phase importante de l'occupation d'une ressource, est associé un événement. Ce qui signifie que plusieurs d'entre eux peuvent survenir entre deux changements d'état d'un processus simulé. Ce processus pris en compte au niveau du modèle dépend directement de la schématisation du workload ; c'est également elle qui sera à la base de la génération des événements. En effet, elle a pour rôle de fournir au modèle, sous une forme quelconque, un ensemble de séquences d'utilisation de ressources simulées. Dès lors, le déroulement de la simulation doit se contenter d'associer les ressources aux processus et de dégager, lorsqu'il est opportun de le faire, l'événement caractérisant une phase déterminée.

(1) voir Mac Dougall [9]

A) Catégories d'événements

La distinction qui va suivre doit être considérée comme purement logique, en ce sens qu'elle n'implique pas nécessairement l'utilisation de ressources différentes.

1) L'événement caractérisant un traitement purement interne.

Lorsqu'un processus simulé a obtenu le CPU pour un traitement déterminé, on génère un événement "fin d'utilisation de CPU" lorsque, pour une raison de préemption, de quantum de temps alloué ou d'entrée-sortie par exemple, la phase de traitement interne exécutée par ce processus est terminée ou stoppée. Il est à remarquer que, tout au long de cette phase, le processus est sensé exécuter du code et faire appel à des données existant en mémoire centrale uniquement. On envisage donc ici un type de phase où n'intervient, à l'intérieur de celle-ci, aucun contexte d'entrée-sortie.

2) Les événements associés à une opération I/O.

Outre l'utilisation exclusive du CPU, il existe des demandes d'entrée-sortie à prendre en compte dans le modèle de simulation. La façon dont chacune de ces demandes peut être représentée a déjà été examinée. Le schéma de la page 76 , repris ci-dessous, apporte la réponse quant aux événements qu'il faut distinguer dans la réalisation d'une opération I / O simulée :

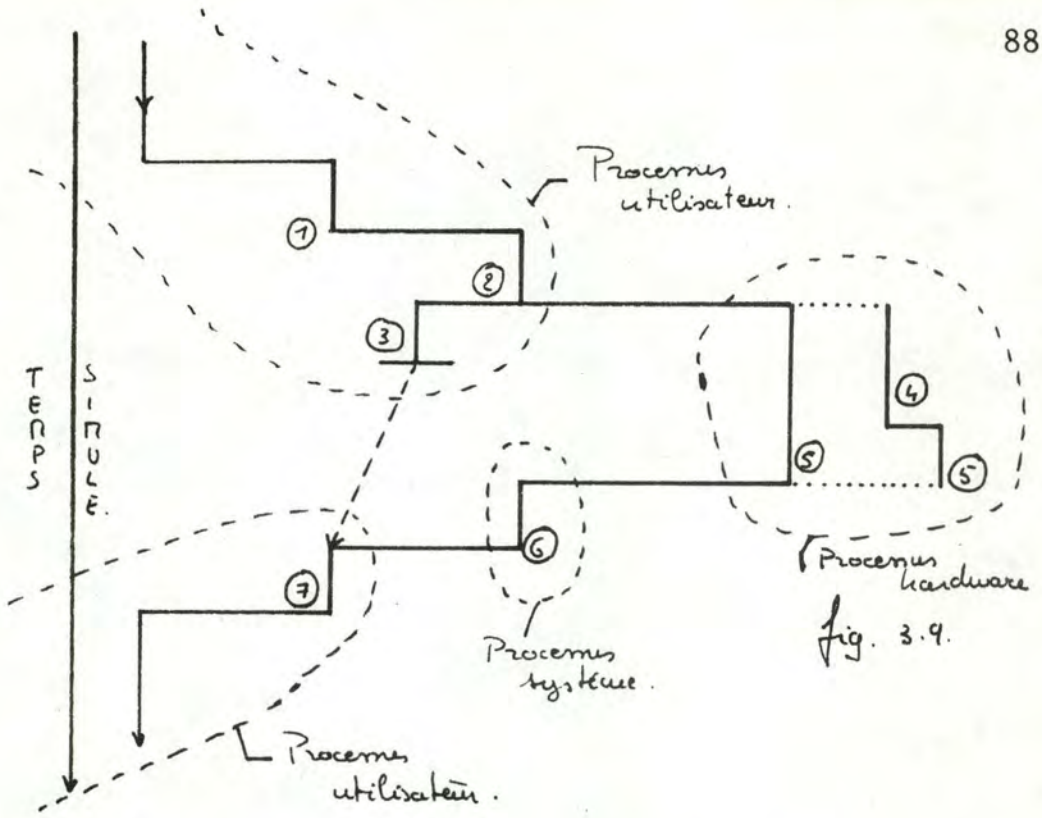


fig. 3.9.

A l'intérieur même des processus simulés, on remarque une ou plusieurs phases de réalisation :

a) <u>le processus utilisateur</u> : il intervient naturellement au début et à la fin de l'opération I / O simulée ; la ressource qu'il utilise est le CPU et les événements que l'on peut distinguer sont les suivants :
   - événement n° 1 : signale une phase du Data Management simulé.
   - événement n° 2 : signale la phase d'initialisation physique de l'I / O.
   - événement n° 3 : signale de nouveau une phase du Data Management simulé et la suspension du processus - utilisateur.
   - événement n° 7 : signale la clôture logique de l'opération I / O après le réveil du processus-utilisateur simulé.

b) le processus hardware : il réalise la simulation de l'opération de transfert des données, précédée s'il y a lieu d'une opération de recherche (notamment pour le disque). Dans ce dernier cas, on distingue deux phases principales d'exécution, signalées chacune par deux événements différents :

- événement n° 4 : signale la phase de recherche simulée sur le dispositif de mémoire secondaire.
- événement n° 5 : signale la phase de transfert simulé des données, à partir de ou vers la mémoire auxiliaire.

Dans le premier cas, où seule la phase de transfert existe, l'événement n° 4 n'aura pas lieu, seul l'événement n° 5 interviendra.

c) le processus système : il réalise la simulation de la clôture physique de l'entrée-sortie. La ressource utilisée est le CPU et on ne distingue qu'une phase (signalée par l'événement n° 6) dans le traitement que le processus effectue.

B) Le problème du parallélisme des processus.

Un simulateur utilisant une horloge pour le temps simulé et admettant l'existence de plusieurs processus parallèles doit résoudre un problème inhérent à la gestion des différents événements, et conséquemment à la progression des processus simulés.

Un exemple où ce problème se pose peut être tiré de la fig. 3.9 : lorsque l'événement ② apparaît, le processus utilisateur d'une part continue à consommer quelque peu du temps simulé et le transfert des données d'autre part (si on suppose que l'événement ④ n'intervient pas) doit être pris en compte à

partir de ce moment également. Il s'ensuit ce qu'on pourrait appeler une concurrence d'événements futurs à laquelle le modèle doit apporter une solution.

Elle pourrait avoir le visage suivant :

- parmi les ressources on distingue du CPU celles pour lesquelles le processus qui les utilise est non-interruptible une fois qu'il est lancé.
- pour ces ressources, on peut associer un moment absolu dans le temps simulé à l'événement qui signalera la fin d'une phase du processus initialisé.
- quant aux processus utilisant le CPU, on leur affecte des tranches de temps dont la grandeur est également absolue, mais dont la venue à terme est relative, étant donné l'interruptibilité des processus simulés et le caractère partageable de la ressource.

La génération des différents événements relève alors du scheduling appliqué.

- - - - -

3.3. GESTION SIMULEE DES RESSOURCES

Après avoir examiné les principales exigences d'une interruption de la gestion des processus au niveau de la simulation, nous allons nous intéresser à certains problèmes que pose la gestion simulée des ressources. Lors d'une exploitation réelle, le système assure le contrôle simultané des uns et des autres selon un ensemble de règles déterminées. De la même manière que pour les processus, il est inutile de reproduire en simulation tous les éléments qui interviennent dans la gestion des ressources, étant donné qu'un avantage primordial d'une méthode de simulation est précisément de laisser au responsable la possibilité de simplifier le mécanisme étudié. Selon l'intérêt et l'objectif que poursuit l'analyste de système, l'évaluation de la gestion des ressources va donc prendre un certain visage. Il n'est nullement question d'aborder ici tous les aspects d'une telle étude mais plutôt de passer en revue les ressources utilisées le plus couramment et d'examiner plus ou moins en détail les nécessités à satisfaire au niveau du modèle pour garder au système simulé les lignes de force et les principales caractéristiques du système réel.

3.3.1. La ressource CPU

La gestion de cette ressource, ou en d'autres termes la stratégie d'allocation de l'organe central de traitement aux différents processus a déjà été évoquée lors du sous-chapitre précédent. Il est donc inutile de reprendre ici les paragraphes antérieurs.

3.3.2. La ressource MEMOIRE CENTRALE

Dans un contexte de multiprogrammation, la gestion de cette ressource est très importante étant donné qu'elle constitue par l'intermédiaire de son résultat, une condition nécessaire au déroulement d'un certain nombre de processus simultanément. Les techniques d'allocation sont nombreuses, et il n'est pas possible de les envi-

sager toutes ici dans un but de comparaison. Parmi elles, une technique de mémoire virtuelle a été retenue ; la suite de ce paragraphe s'attachera à cerner les exigences de sa représentation en simulation, et ceci dans la même optique que celle qui a été choisie depuis le début de cette étude. Il est utile avant de poursuivre, de rappeler les caractéristiques principales liées à une mémoire dite virtuelle :

1) pour un utilisateur donné, l'espace des noms engendré par l'ensemble des programmes qu'il soumet à l'exécution et donc au contrôle du système, est potentiellement beaucoup plus grand que l'espace d'adresses dont il dispose en mémoire.
2) une conséquence immédiate du point 1) oblige le corps des programmes qui ne peuvent se trouver en mémoire principale à être stockés de manière temporaire sur un dispositif de mémoire auxiliaire.
3) à tout moment de l'exécution du programme de l'utilisateur, peut se produire ce que l'on appelle une "faute de page" ou une "faute de segment", selon l'organisation implémentée.
4) il est donc nécessaire de prévoir un dispositif qui effectuera la recherche de l'information ou du code absents de la mémoire centrale, après qu'un algorithme de remplacement ait trouvé un espace suffisant à l'intérieur de celle-ci.

L'organisation de la mémoire virtuelle dont il sera question dans les quelques pages qui suivent repose sur le concept de segment (ou entité logique) et ne suppose pas l'existence de pages (entités physiques).

A) Conséquences d'une optique "simulation"
.....

Il est clair qu'à l'intérieur d'un operating system, la fonction qui gère la mémoire virtuelle occupe une place non négligeable et peut présenter un degré de

complexité respectable. En raison de sa nature même, elle couvre une série de domaines qui nécessitent déjà chacune un traitement assez important :

- la gestion de la mémoire centrale.
- la gestion des segments.
- la gestion de la mémoire secondaire.
- la gestion des exceptions (fautes de pages, fautes de segments).

L'objectif qui tend à réaliser une étude des performances d'un tel dispositif par la simulation peut difficilement porter d'emblée sur l'entièreté des fonctions existantes.

Selon le type de résultat voulu, le modèle reprendra dans une première étape la simulation d'un aspect de la mémoire virtuelle pour lequel l'estimation des performances apparaît comme plus importante qu'ailleurs ; toutefois, la possibilité d'intégrer d'autres fonctions dans la suite doit être sauvegardée.

En ce qui nous concerne, c'est la représentation en simulation d'un algorithme de remplacement qui sera examinée sous l'angle des éléments à introduire dans le modèle, selon le niveau de détail que l'on veut assurer à l'estimation et par conséquent, selon l'éventail des résultats espérés.

Etant donné que l'organisation de la mémoire virtuelle envisagée est à base de segments, ceux-ci doivent intervenir comme entités à prendre en compte dans le modèle d'évaluation. Néanmoins, une restriction déjà apportée auparavant en ce qui concerne les éléments traités par la simulation, doit de nouveau être présente dans ce contexte-ci : les segments n'existent pas en simulation, et par conséquent doivent intervenir via un intermédiaire, qui peut être constitué par un descripteur. Celui-ci, au même titre qu'un descripteur de processus, est chargé de signaler à tout moment au

modèle l'état de l'entité qu'il représente. Selon le niveau de détail auquel on veut prendre les segments en considération, on intègre dans le descripteur les informations suivantes :

- l'adresse du segment en mémoire centrale.
- la taille du segment.
- un indicateur de présence ou d'absence du segment en mémoire centrale.
- un indicateur de stabilité du segment, qui traduit la possibilité ou l'impossibilité d'accéder à ce segment.
- un indicateur d'utilisation du segment.
- un indicateur de modification interne du segment.
- l'adresse du segment en mémoire secondaire.
- un facteur de résidence du segment en mémoire centrale.
- le type du segment (cette information sera précisée dans la suite).
- la possibilité de partager le segment ou non.
- la possibilité éventuelle de reloger le segment.
- la possibilité éventuelle d'enlever le segment de la mémoire centrale.

La segmentation constitue une opération qui permet une gestion de la mémoire sur base d'entités logiques, au contraire de la pagination qui implique des entités physiques. De ce fait, l'utilisateur est en mesure de spécifier au système un ensemble d'attributs au moyen desquels il caractérise les segments qui font partie du workload. La conséquence immédiate de ceci est la suivante : si le modèle a pour but d'estimer l'efficacité d'une mémoire virtuelle segmentée, ou tout au moins celle d'une fonction parmi celles qui composent un tel dispositif, il est naturel d'introduire dans l'input du simulateur les caractéristiques des segments que l'on suppose inclus dans la charge de travail.

B) La procédure de simulation

1) Préliminaires

Comme signalé ci-dessus (cf. page 93), l'examen des nécessités à satisfaire et des possibilités qui existent pour intégrer dans un modèle de simulation une fonction choisie parmi toutes celles qui constituent la gestion d'une mémoire virtuelle, va porter sur un algorithme de remplacement comprenant quatre phases dont le rôle sera expliqué dans la suite.

Avant d'aborder cet algorithme, il est nécessaire de préciser quelques éléments qui interviennent à l'intérieur de celui-ci, et qui ont trait soit aux segments eux-mêmes, soit à des espaces de mémoire centrale.

On appelle :

*) segments de type A : un segment est de type A au temps T simulé si à ce moment T, il ne peut être soumis à une procédure de swapping.

*) segments de type B : un segment est de type B au temps T simulé si à ce moment T, il peut être soumis à une procédure de swapping.

Cette information complète la notion de type de segment, signalée en p. 94.

On appelle :

*) espace libre de type 1 : un espace mémoire qui au temps T simulé, n'est requis par aucun processus et ne comporte donc aucun segment auquel on puisse faire référence.

*) espace libre de type 2 : un espace mémoire qui, au temps T simulé, est composé d'une suite continue comprenant des segments de type B et des espaces libres de type 1 en nombre et ordre quelconques.

*) espace libre de type 3 : un espace mémoire qui, au temps T simulé, est composé :
- d'un et d'un seul segment de type A,
- contigu à un ou deux espaces libres de type 2.

Des exemples de chacun de ces éléments sont donnés par la fig. 3.10

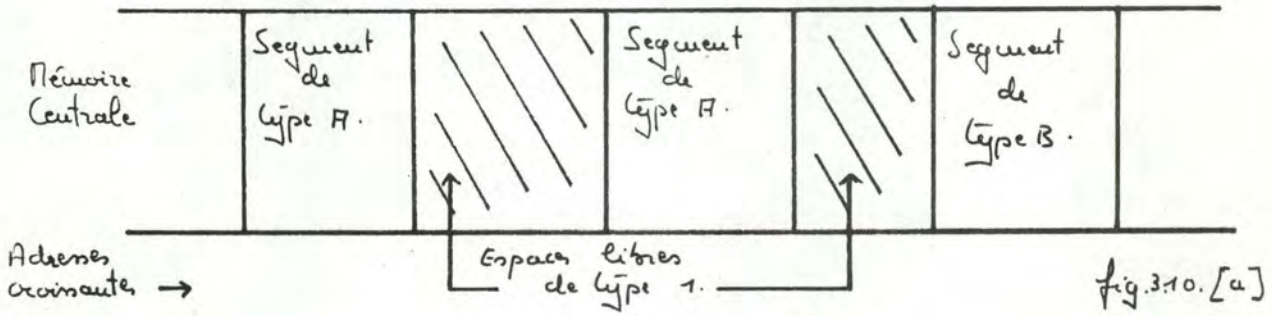


fig. 3.10. [a]

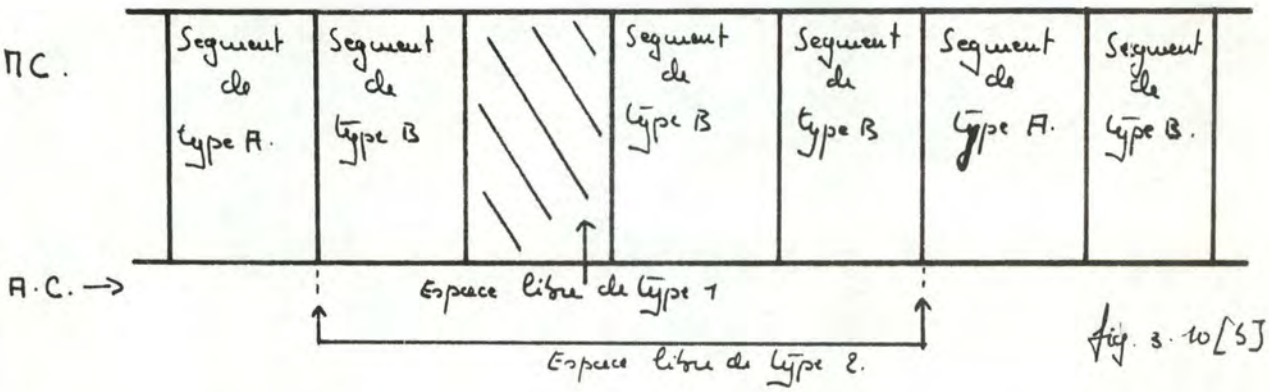


fig. 3.10 [b]

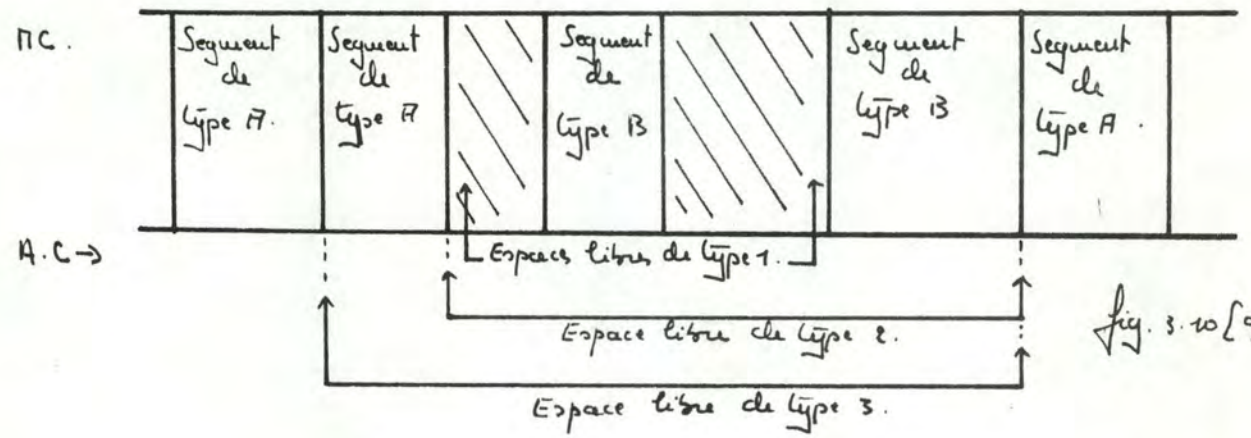


fig. 3.10 [c]

2) L'algorithme de remplacement

Dès que le processus utilisateur fait appel à un segment qui à ce moment ne se trouve pas en mémoire centrale, une exception "défaut de segment" est générée. Les mécanismes de la gestion de la mémoire virtuelle entrent en jeu, et l'algorithme de remplacement doit trouver en mémoire virtuelle un espace suffisant pour recevoir le segment manquant. Il comporte 4 phases :

- Phase 1 : elle tente de trouver un espace libre de type 1 appelé à recevoir le segment.
- Phase 2 : en cas d'insuccès de la phase 1, la phase 2 tente de trouver un espace libre de type 2 suffisant.
- Phase 3 : en cas d'insuccès de la phase 2, la phase 3 tente de trouver un espace libre de type 3 suffisant.
- Phase 4 : en cas d'insuccès de la phase 3, la phase 4 réalise la compaction des segments.

3) Comment envisager la simulation

Il est possible d'envisager la simulation de l'algorithme de deux manières différentes :

- la première est basée sur le hasard et génère des fautes de segment ainsi que les passages dans les différentes phases de manière aléatoire.
- la seconde s'éloigne quelque peu de l'aspect simulation pure et reproduit les phases de l'algorithme, qui seront exécutées s'il y a lieu de le faire.

a) Première méthode

Pour qu'elle puisse être adoptée, il est nécessaire que le modèle connaisse a priori un certain nombre de valeurs, qui ne peuvent être obtenues généralement que par des résultats de mesures :

- la probabilité qu'un appel à un segment génère un défaut de segment.

- la probabilité de suivre les différents chemins à l'intérieur de l'algorithme.
- pour chacun de ces chemins, une distribution des temps nécessaires à leur parcours.

Connaissant ces éléments, le modèle peut alors, dès que la schématisation du workload fait apparaître un appel à un segment, décider de façon probabiliste s'il existe un défaut de segment ; dans le cas affirmatif, il détermine également de façon aléatoire le parcours qu'il est nécessaire de simuler à l'intérieur de l'algorithme, ainsi qu'une durée qu'on affecte à ce parcours.

Lorsque selon les valeurs obtenues, la simulation du mécanisme est achevée, le modèle considère que le segment auquel on avait en vain fait référence auparavant se trouve en mémoire centrale, et que dès lors le processus utilisateur simulé, après avoir été dévié sur une fonction de la mémoire virtuelle, peut continuer son traitement normal, qu'il reprend là même où l'absence simulée du segment l'avait empêchée de poursuivre.

Sur le plan de la gestion des événements, et toujours par application du principe qui associe un événement à toute phase logique intervenant dans la vie d'un processus simulé, on peut prévoir :

- un événement signalant la génération d'un défaut de segment.
- un événement signalant le début de la simulation de l'algorithme de remplacement.
- un événement signalant la fin de la simulation du chemin parcouru à l'intérieur de l'algorithme ; on associe un événement particulier à chaque chemin possible, ce qui implique quatre événements différents.
- un événement signalant la présence simulée en mémoire centrale du segment référencé comme

absent auparavant ; il indique également la suite du processus utilisateur simulé.

Cette méthode présente l'avantage d'être plus courte que la suivante ; par contre, elle nécessite pour sa mise en oeuvre une connaissance déjà précise du comportement de l'algorithme et de la fréquence des exceptions générées pour faute de segment.

b) Seconde méthode

1. Eléments de base

Alors que la première méthode fait un usage nul du descripteur de segment, la seconde utilise celui-ci de façon intensive. Elle a pour objectif de gérer elle-même une mémoire centrale fictive d'une dimension donnée, composée en nombre et ordre quelconques, de segments de type A et B, ainsi que d'espaces libres de type 1. Il est dès lors important de donner au modèle les éléments de base nécessaires à la gestion qu'il doit effectuer ; ces éléments sont constitués par les descripteurs des segments qui sont supposés se trouver en mémoire principale, auxquels il faut ajouter, pour couvrir tout l'espace mémoire dont on est sensé disposer, les "descripteurs" d'espaces libres de type 1.

L'introduction d'un descripteur pour les espaces libres, bien qu'elle paraisse bizarre à première vue, présente toutefois l'avantage pour le modèle de disposer d'un élément entièrement standardisé qui tantôt représente un segment, tantôt représente un espace libre de type 1.

Il est clair que l'information concernant un segment est beaucoup plus fournie que celle qui a trait à un espace libre. On se contentera

pour ce dernier de reprendre les caractéristiques suivantes :

- son adresse début en mémoire centrale.
- sa taille.
- un indicateur d'espace libre, dont le rôle peut être assimilé à celui du type de segment.

De plus, étant donné que le contexte est celui d'une mémoire virtuelle, il est vital de tenir compte des segments qui au temps T simulé, ne se trouvent pas en mémoire principale, mais sont stockés sur mémoire secondaire.
Par conséquent, l'ensemble des descripteurs que le modèle doit prendre en compte est composé des constituants suivants :

- les descripteurs d'espaces libres de type 1 (voir I ci-dessous).
- les descripteurs de segments de type A (voir II ci-dessous).
- les descripteurs de segments de type B (voir III ci-dessous).
- les descripteurs des segments stockés en mémoire secondaire (voir IV ci-dessous).

La figure 3.11 intègre tous ces éléments en faisant appel à la notion d'intersection d'ensembles.

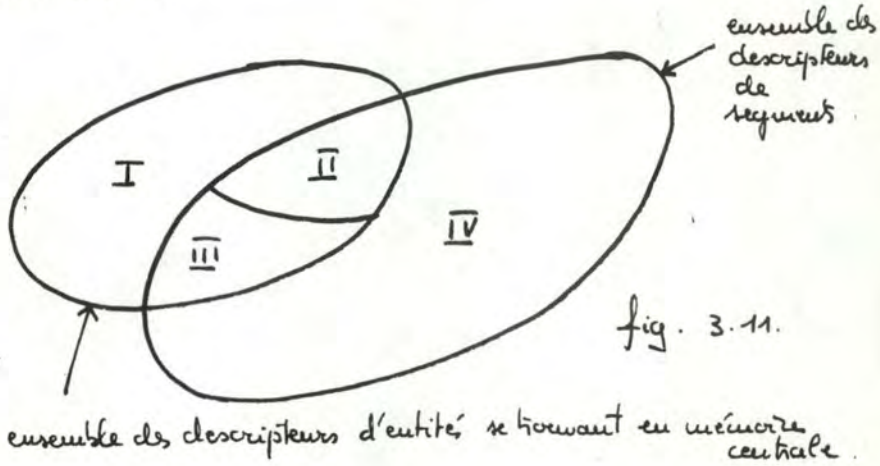


fig. 3.11.

De façon à être en mesure de pouvoir rechercher un descripteur donné à tout moment, le modèle doit prévoir une organisation qui lui permette de satisfaire certaines exigences.

En effet, il doit lui être possible :

- de retrouver ensemble tous les descripteurs des entités (segments et espaces libres) qui constituent le contenu de la mémoire centrale (zones I, II, III de la fig. 3.11).
- de retrouver, pour chaque zone séparée appartenant à la même figure, les descripteurs d'entités qui la constituent.

En conséquence, il est commode, pour répondre à ces nécessités, de prévoir autant de listes qu'il existe de critères selon lesquels on désire atteindre les descripteurs.

2. La procédure de simulation

Etant donné que le modèle gère lui-même la mémoire centrale fictive, il connaît en permanence l'état de celle-ci et est en mesure de déterminer, au vu de cet état, si une référence à un segment donné, présente dans la schématisation du workload, peut être simulée au moment où elle est sollicitée. En effet, par l'intermédiaire de l'indicateur de présence introduit dans le descripteur, il est possible de déceler l'existence ou l'absence simulée du segment en mémoire principale.

En conséquence, lorsque en cours d'exécution du modèle, à l'intérieur d'un traitement simulé utilisant le CPU, un appel à un segment donné est provoqué, une alternative se présente :

- si le segment est considéré comme présent en mémoire centrale et dès lors, la simulation du processus se poursuit normalement.

- le segment est considéré comme inexistant en mémoire centrale. Il est nécessaire dans ce cas de suspendre le traitement interne simulé et d'exécuter la procédure dont l'objectif est de trouver un espace fictif en mémoire centrale, destiné à contenir le segment manquant.

  Lorsque ceci est réalisé, l'indicateur de présence est mis à jour dans le descripteur adéquat et le traitement suspendu auparavant peut recommencer.

C'est de façon globale la procédure à implémenter dans le modèle de simulation. Le cas où le processus - utilisateur simulé poursuit sa vie de façon normale ne nécessite pas de développement supplémentaire, étant donné que la seule modification par rapport à une version du modèle où n'existerait pas l'étude de la mémoire virtuelle, consiste à exécuter un test sur l'indicateur de présence. Par contre, le cas d'absence du segment voulu implique un traitement particulier dans le modèle, dont les principes sont exposés ci-après.

La procédure doit tenir compte des problèmes suivants :

- lors de la reconnaissance de l'absence simulée en mémoire centrale, l'événement qui aurait dû caractériser la phase logique future du processus simulé doit être mis temporairement hors-circuit. Il reviendra en ligne de compte lorsque le segment manquant sera considéré comme présent en mémoire principale.
- la procédure elle-même étant protégée, il est hors de question que deux processus simulés ou plus l'utilisent en même temps. Par conséquent, si un second processus simulé P' échoue sur une faute de segment alors qu'un premier processus simulé P bénéficie à ce moment des services de l'algorithme de remplacement, il est nécessaire de suspendre le processus P' en attendant que le processus P puisse

libérer la procédure, qui est en fait considérée comme une ressource non partageable entre plusieurs processus simulés simultanés.

En admettant que le modèle dispose en permanence de la valeur de la taille du plus grand espace libre de type 1 ainsi que du même renseignement pour les espaces libres de type 2, on établit comme suit (fig.3.12) le schéma global de l'algorithme tel qu'on peut l'introduire dans le modèle :

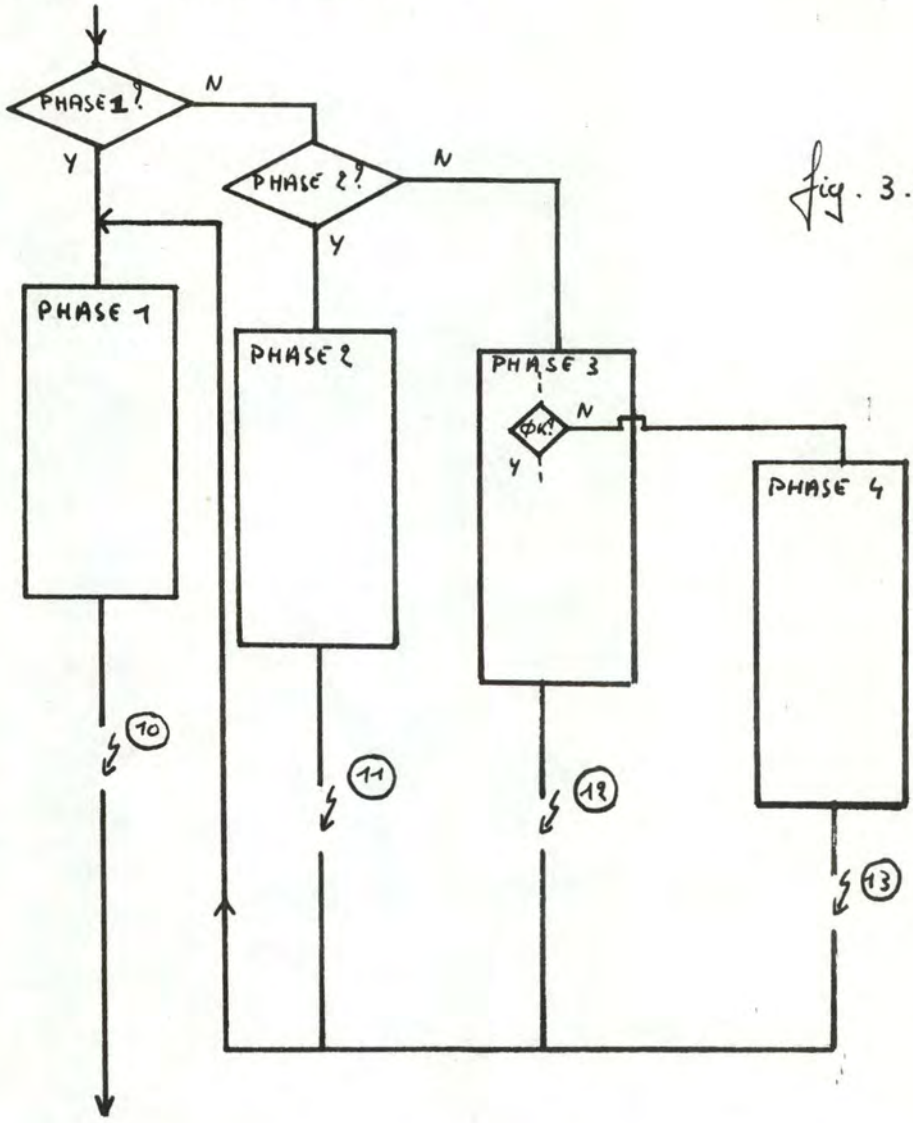


fig. 3.12.

Lorsque la procédure a résolu les deux problèmes mentionnés ci-dessus, à savoir la suspension de l'événement futur et le blocage d'éventuels processus simulés, elle s'attache à exécuter les fonctions

de l'algorithme de remplacement.

La fig. 3.12 schématise ces fonctions :

- la valeur de la taille du plus grand espace libre de type 1 indique s'il y a lieu ou non d'exécuter d'emblée la phase 1.
  Cette phase met à jour la taille de l'espace libre, ainsi que le descripteur du segment pour lequel on va simuler un peu plus tard une opération de transfert de la mémoire secondaire vers la mémoire principale.
- si la phase 1 ne peut pas encore être exécutée, la valeur de la taille du plus grand espace libre de type 2 indique s'il y a lieu d'exécuter la phase 2. Dans le cas affirmatif, cette phase recherche le premier espace libre de type 2 qui puisse accueillir le segment manquant. S'il y a lieu, elle simule des opérations de transfert de la mémoire principale vers la mémoire secondaire pour les segments de type B qui sont supposés avoir été modifiés depuis leur dernière entrée simulée en mémoire principale.
  Dans le cas négatif, on passe à la phase 3.
- la phase 3 recherche un espace libre de type 3 qui puisse être retenu pour contenir le segment manquant. Dans le cas où cette recherche est couronnée de succès, la phase simule si du moins c'est nécessaire un transfert de sortie vers la mémoire auxiliaire pour chaque segment de type A et B qui est supposé avoir été modifié lors de son dernier séjour en mémoire centrale.
- la phase 4 est exécutée en cas d'insuccès de la recherche exécutée dans la phase 3 et a pour but d'enlever de la mémoire centrale des segments de type B, et de type A ensuite si nécessaire, jusqu'au moment où une compaction des segments restants permet au segment manquant de se loger en mémoire centrale.

- chacune des phases 2, 3, 4 a pour effet de trouver un espace libre suffisant auquel on puisse donner, lorsqu'elle est exécutée, le type 1. Il est donc nécessaire avant toute chose de mettre à jour la taille de ce nouvel espace libre ; c'est la raison du retour en phase 1 après l'exécution d'une des autres phases.

Note : Il est utile de rappeler ici que toutes les opérations décrites ci-dessus et mettant en jeu les segments, traitent en réalité leur descripteur uniquement.

Lorsque la procédure a trouvé, au moyen de l'algorithme de remplacement, et mis à jour un espace suffisant pour le segment manquant, il est encore nécessaire :

- de simuler une opération de transfert de ce segment vers la mémoire centrale.
- de restaurer pour le processus utilisateur simulé P le n° d'événement qui a fait l'objet d'une mise hors-circuit en début de procédure.
- de réveiller un éventuel processus simulé P' qui aurait échoué sur un défaut de segment avant que le processus P n'ait pu disposer du segment qui lui manquait.

3. Les événements

Les différents événements qu'on peut associer aux différentes phases logiques du traitement d'un défaut de segment sont les suivants :

- un événement signalant la reconnaissance d'une absence de segment et le début des opérations voulues appartenant à la gestion de la mémoire virtuelle.
- un événement signalant le début de l'exécution de l'algorithme de remplacement.
- pour chacune des phases de l'algorithme, un événement signalant la fin de l'exécution de la

phase. Les événements 10, 11, 12, 13 introduits dans la fig. 3.12 répondent à ce besoin.

- un événement particulier signalant la fin de la simulation d'une entrée-sortie nécessitée pour la procédure de gestion et la mémoire virtuelle.

4. Signification et but de la méthode

Au contraire de la première méthode exposée, on constate que celle-ci :

- ne nécessite pas une connaissance de la fréquence des défauts de segments. En effet, le principe appliqué dans la seconde méthode vise à faire déterminer d'une part par la schématisation du workload et d'autre part par l'analyse de l'état de la mémoire centrale fictive, les occasions où peut se présenter une absence de segment en mémoire principale (1).
- ne nécessite pas non plus la connaissance des probabilités de parcours des différents chemins à l'intérieur de l'algorithme de remplacement. De nouveau, c'est l'analyse de l'état de la mémoire centrale fictive qui déterminera la(les) fonction(s) de l'algorithme qui sera(seront) effectivement exécutée(s).

Cependant, des distributions de valeurs des temps requis pour le parcours des différents chemins lors de l'exécution réelle de l'algorithme implémenté dans le software sont toujours nécessaires, étant donné que cela n'aurait aucun sens de se servir du temps d'exécution du modèle de simulation. Au vu de tout ceci, la signification et le but de la méthode apparaissent comme étant la volonté, à travers un modèle de simu-

(1) La constatation d'une faute de segment est liée à l'examen du contenu du working-set associé au processus simulé ; toutefois, il n'entre pas dans le cadre de ce travail de préciser la façon dont ce working-set peut être envisagé.

lation, de gérer et surtout de pouvoir contrôler à tout moment l'état dans lequel se trouve la mémoire centrale. Au lieu de s'en remettre aveuglément à une procédure générant des valeurs aléatoires et ne donnant comme résultat qu'un état final, on veut pouvoir suivre pas à pas le déroulement et le comportement d'une fonction simulée.
Il est également intéressant de noter que cette approche s'inscrit assez naturellement dans le même ordre que celui de la démarche effectuée pour obtenir un input du modèle ; de la même manière en effet, on veut aboutir à une représentation fidèle du réel.

3.3.3. Les canaux et les mémoires auxiliaires

La gestion de ces ressources doit elle aussi en simulation s'inspirer des mécanismes réels pour assurer, au niveau d'intérêt auquel on se place, une représentativité valable au sein du modèle. Selon le degré d'intervention que l'on veut donner à chacune de ces ressources, et selon le rôle qu'on entend leur affecter dans le modèle, la complexité du traitement qui leur est relatif sera plus ou moins grande.

A) Les canaux d'entrée-sortie

Lors d'une demande d'entrée-sortie, générée de façon aléatoire ou précisée par le workload, il est nécessaire avant de pouvoir simuler l'opération, de contrôler la disponibilité du canal qui permet l'accès au dispositif de mémorisation concerné.
Ce problème peut se résoudre de façon aisée par l'association à chaque numéro de canal d'un indicateur de disponibilité qui puisse être testé et modifié selon la nécessité.

Etant donné le niveau parfois élevé de multiprogrammation, et la relative lenteur des I/ O par rapport

au traitement CPU, il est utile de prévoir la possibilité d'une file d'attente devant chaque canal.

B) Les disques

Ces dispositifs possèdent un certain nombre de caractéristiques relatives aux temps d'accès ainsi qu'aux temps de transfert des informations. La simulation doit prendre en compte ces caractéristiques en les intégrant dans des combinaisons différentes selon l'organisation du fichier auquel on est supposé accéder.
A côté de cet aspect du temps, le modèle doit aussi, lors d'une entrée-sortie, considérer la disponibilité ou l'indisponibilité du dispositif. Dans ce dernier cas, le processus simulé qui demande l'I/ O entre dans la file d'attente adéquate.
Il se peut que l'étude de simulation n'ait pas pour but, dans une première étape, de pouvoir dégager certaines règles relatives p. ex. à l'occupation optimale du disque. Néanmoins, le modèle doit être structuré de façon à ce qu'un tel projet soit aisément réalisable dans la suite ; ce pourrait être le cas si les responsables décident de simuler, pour une mémoire virtuelle, une gestion de la mémoire centrale et du disque dans le but de dégager des critères d'optimalité.

C) Les bandes magnétiques

Au niveau du modèle, les caractéristiques des temps d'accès et de transfert peuvent intervenir selon le niveau de détail que l'on souhaite introduire dans la simulation d'une entrée-sortie. De nouveau, la vérification de la disponibilité du dispositif est nécessaire, et la possibilité de construire une file d'attente doit être retenue.

D) Le lecteur de cartes et l'imprimante

Le test de disponibilité doit toujours être présent ; quant aux temps de transfert, on peut considérer en simulation qu'ils sont à peu près constants.

*

* *

C H A P I T R E IV :

OUTPUTS DU MODELE D'EVALUATION

4.1. Rappel de l'objectif

4.2. Catégories d'outputs

4.3. Problème de l'acquisition des résultats

4.1. RAPPEL DE L'OBJECTIF

L'évaluation des performances d'un O.S. a été envisagée, au cours de cette étude, sous l'angle des nécessités à prendre en compte lors de l'élaboration d'un modèle de simulation. Les lignes de force qui composent celui-ci ont pour but de constituer une structure et un contenu qui permettent de dégager, à posteriori dans une première étape, à priori dans la suite, les normes de comportement d'un système d'exploitation dont on connaît les fonctions et caractéristiques importantes.

Il était hors de question d'examiner ce qu'aurait pu être la représentation d'un operating system entier à l'intérieur d'une simulation ; en conséquence, parmi les nombreuses fonctions réalisées dans un système, un choix était nécessaire. Il s'est porté plus particulièrement sur la gestion des processus et la gestion des ressources, considérées toutes deux dans certaines limites. Le but de ce chapitre est de tenter de déterminer, sans toutefois prétendre à l'exhaustivité, la nature des informations susceptibles de présenter un intérêt pour l'analyse de l'efficacité d'un système. Cette analyse, si l'on se réfère au schéma de la page 24 , doit aboutir à un ensemble de conclusions sur base desquelles la poursuite de l'évaluation sera ou non envisagée, et de quelle manière. En effet, en fonction des résultats obtenus, l'étude peut se prolonger de diverses façons:

1) la phase d'évaluation est stoppée, et on met à jour éventuellement la version de l'O.S.
2) on décide de modifier l'O.S. et par conséquent le modèle de simulation, de façon que celui-ci soit conforme au produit évalué.
3) seul le modèle est concerné par une modification, qui peut être réalisée soit dans le cadre des fonctions déjà représentées, soit par l'inclusion d'autres fonctions du système.

Un exemple illustrera ces notions de façon plus précise : il se peut que l'analyse des résultats de la simulation révèle l'inopportunité d'une phase dans l'algorithme de remplacement des segments ; dans ce cas, on peut décider de supprimer la représentation de cette phase dans le modèle et d'examiner les conséquences de cette suppression.

Note. Il est bien entendu que ceci n'est donné qu'à titre exemplatif ; il n'entre pas dans le cadre de ce travail de procéder à l'analyse des résultats d'un modèle.

- - - - -

4.2. CATEGORIES D'OUTPUTS

4.2.1. Outputs relatifs à la gestion simulée des processus.

La manière dont on a envisagé la construction de l'input du simulateur est basée sur le principe d'une représentation valable des applications réelles que l'on peut soumettre au contrôle du système. Il est donc normal de pouvoir tirer des conclusions lors de l'output du modèle, en fonction de ce que celui-ci a reçu comme base de travail.

Les critères de performances naturels pour l'utilisateur, qui sont par exemple le turnaround time et le throughput peuvent être considérés en fin de simulation, et les résultats relatifs à ces critères peuvent être déduits facilement :

- le turnaround time correspond à la valeur du temps simulé qui fut nécessaire à l'"exécution" d'une application particulière à l'intérieur du modèle ; il est possible de relever l'heure de début et de fin de simulation et par conséquent d'établir pour chaque application une estimation du temps qu'elle nécessiterait pour être exploitée dans le contexte réel.
- le throughput est donné par le nombre d'applications différentes qu'il est possible de simuler entièrement pendant une tranche de temps donnée.

Un critère souvent cité est celui de la disponibilité du CPU. Celle-ci peut être mesurée aisément, étant donné que le modèle a le contrôle complet de l'allocation et de la reprise du CPU à chaque processus simulé.

Les résultats suivants concernent plus spécialement la gestion des processus eux-mêmes, et de ce fait caractérisent plus directement le comportement du système.

Il a été établi précédemment qu'étant donné d'une part le degré parfois élevé de multiprogrammation et d'autre

part, le nombre limité de ressources, celles-ci peuvent être sollicitées par plusieurs processus simultanément. Il en résulte une création de files d'attente que l'on peut répertorier :

- selon l'état de chacun des processus dont elles contiennent le descripteur,
- et selon la ressource que ceux-ci attendent.

Le résultat d'un tel classement est constitué par :

- une ou plusieurs files d'attente pour les processus "prêts" à utiliser le CPU.
- plusieurs files d'attente pour les processus "suspendus", bloqués sur un accès à un fichier donné.
- une file d'attente des processus devant chaque canal.
- une file d'attente des processus devant chaque dispositif d'entrée-sortie.

Des valeurs statistiques peuvent être élaborées sur plusieurs bases différentes : l'une concernant chaque file d'attente, les autres concernant chaque processus simulé.

1° Pour chaque file d'attente on peut retenir :

1) sa longueur minimum atteinte au cours de la simulation.
2) sa longueur maximum atteinte au cours de la simulation.
3) sa longueur courante (pendant un intervalle de temps).
4) sa longueur moyenne, calculée comme suit :

$$\text{long. moyenne} = \frac{\text{Somme des longueurs courantes}}{N}$$

avec N = nombre d'observations de la longueur courante (une observation est réalisée à chaque occurrence d'un événement).

5) sa longueur moyenne pondérée par le facteur temps, et qui s'établit comme suit :

$$\text{longueur moyenne pondérée} = \frac{\sum_{i=1}^{m} (l_i * t_i)}{\sum_{i=1}^{m} t_i}$$

avec m = nombre d'intervalles de temps (sur la durée totale du temps simulé).

l_i = longueur courante de la liste pendant l'intervalle n° i.

t_i = valeur en temps simulé de l'intervalle n° i.

6) la distribution de ses différentes longueurs au cours de la simulation. Chaque couple (a, b) de cette distribution est obtenu de la façon suivante :

*a = longueur de la file d'attente

*b = fraction du temps pendant laquelle la file d'attente a été de longueur égale à a.

$$= \left(\sum_{j=1}^{k} t_j \right) / \left(\sum_{i=1}^{m} t_i \right).$$

avec k = le nombre d'intervalles pendant lesquels la file d'attente est de longueur a.

t_j = valeur en temps simulé de l'un de ces intervalles.

m = nombre total d'intervalles de temps.

t_i = valeur en temps simulé d'un intervalle n° i.

2° <u>Pour chaque processus utilisateur simulé on peut retenir</u> :

1) pour la (ou les) file(s) d'attente où il est placé lorsqu'il est dans l'état "prêt" :

- le temps d'attente minimum
- le temps d'attente maximum
- le temps d'attente totale
- le temps moyen d'attente, obtenu par l'expression

$$\left(\sum_{k=1}^{n} t_k \right) / n$$

avec t_k = valeur en temps simulé d'un intervalle de n° k pendant lequel le processus-utilisateur simulé est resté dans la file d'attente CPU.

n = nombre de fois que ce processus-utilisateur est entré dans une file d'attente CPU.

2) le même type de valeurs que celles-ci-dessus, relativement à la (aux) file (s) d'attente où le processus-utilisateur est placé lorsqu'il est dans l'état "suspendu".

3) des valeurs statistiques sur l'occupation du CPU par le processus-utilisateur

- le temps d'occupation minimum
- le temps d'occupation maximum
- le temps total d'occupation
- le temps moyen d'occupation, obtenu par une formule analogue à celle mentionnée ci-dessus.

4) les valeurs $\frac{T_a}{T_t}$, $\frac{T_p}{T_t}$, $\frac{T_s}{T_t}$ avec :

T_t = temps total de la vie du processus-utilisateur simulé

T_a = temps total pendant lequel ce processus a utilisé le CPU.

T_p = temps total pendant lequel ce processus était dans l'état "prêt".

T_s = temps total pendant lequel ce processus était dans l'état "suspendu".

Il est clair que tous les temps simulés relatifs à un processus-utilisateur doivent appartenir à l'intervalle pendant lequel ce processus est en vie.

3° On peut calculer les mêmes types de valeurs, relatives cette fois aux processus-système simulés et aux processus-hardware simulés, en tenant compte du fait que pour eux, seuls existent les états "actif" et "prêt".

Outre ces ensembles de valeurs concernant spécialement les files d'attente et les processus simulés, il doit être possible d'obtenir des renseignements de types suivants :

- le nombre d'allocations du CPU réalisées par le modèle pendant un temps simulé donné.
- un "trace" de la vie de chaque processus.
- si les processus ont une priorité, le nombre de changements de priorité qu'a connu le CPU.

Enfin, pour clôturer ce paragraphe consacré aux outputs "performance" relatifs à la gestion des processus, le modèle doit être en mesure de pouvoir livrer, à n'importe quel instant du temps simulé, le contenu d'une file d'attente quelconque.

4.2.2. Outputs relatifs à la gestion simulée des ressources.

A) Le cas de la Mémoire Centrale

La technique de gestion de mémoire centrale pour laquelle on a examiné au chapitre III les possibilités de simulation est une technique de mémoire virtuelle segmentée, mais non pagée.
Deux méthodes d'intégration d'un tel dispositif dans un modèle d'évaluation de performances y ont fait l'objet d'un développement plus ou moins détaillé ;

à chacune d'elles sont associés des résultats que l'on peut obtenir en cours de simulation ou à la fin de celle-ci, et dont la nature est précisée dans les lignes qui suivent.

1) Outputs relatifs à la première méthode

Ils sont établis en fonction des valeurs aléatoires sur lesquelles est basée la simulation de l'algorithme de remplacement. Ils peuvent se rapporter à chaque processus-utilisateur simulé ou à la simulation dans son entièreté.

a/ Pour chaque processus simulé : on élabore des valeurs statistiques qui, étant donné le contexte dans lequel elles se placent, peuvent également présenter un intérêt au niveau de la gestion des processus.

1) le nombre de défauts de segments générés au cours de la vie du processus simulé.
2) le nombre total de références faites d'un segment à un autre.
3) le rapport des valeurs 1) et 2)
4) le temps simulé total nécessaire aux opérations de mémoire virtuelle.
5) le temps simulé total nécessaire au parcours de chaque type de chemin à l'intérieur de l'algorithme de remplacement.
6) le nombre total d'utilisations, le temps minimum, moyen et maximum de parcours de chacun de ces mêmes chemins.
7) le rapport du temps simulé mentionné en 4 et de la durée totale de la vie du processus.

b/ Pour la simulation dans son ensemble : des valeurs de même type peuvent être calculées ; leur signification reste identique, mais leur domaine de validité est élargi.

2) Outputs relatifs à la seconde méthode

a/ Selon le point de vue "espace". La gestion de la mémoire centrale fictive étant assurée par le modèle lui-même (grâce aux descripteurs d'espace libres de type 1 et de segments de type A et B), il est possible à celui-ci de livrer un ensemble d'informations relatives à l'état de la ressource gérée.

Une carte ou un schéma précisant la nature et les caractéristiques des entités qui composent le contenu de mémoire centrale est souhaitable ; elle permet en effet à l'analyste de système de contrôler en cours la simulation.:

1) le coëfficient de charge de la mémoire.
2) la fréquence de "swap in" et de "swap out" de certains segments, selon leur taille par exemple.
3) le nombre de segments de type A et de type B.
4) le nombre et la taille (min., moyenne, max.) d'espaces libres de type 1, 2, 3.
5) la fréquence des compactions.
6) le nombre de segments appartenant à chaque processus utilisateur.
7) la rapidité de changement de contenu de certaines régions de la mémoire (adresses hautes, adresses basses).

L'intérêt de ceci est de dégager, s'il est possible de le faire, une ou plusieurs tendances qui apparaissent et qui sont relatives à l'occupation et à l'utilisation de la mémoire centrale.

En tenant compte du fait, lors de l'analyse des résultats, que ceux-ci sont obtenus dans le cadre d'un certain algorithme et pour les dis-

tributions de segments donnés, des conclusions peuvent s'imposer ou simplement être suggérées quant à l'efficacité d'une telle structure de mémoire virtuelle et de l'algorithme utilisé pour le remplacement des segments. En fonction de ceci, des modifications peuvent être envisagées sur divers plans, notamment :

- au niveau du modèle lui-même, où on peut exiger un degré de précision plus élevé quant à la représentation de l'algorithme ou des propriétés des segments.
- au niveau de l'algorithme lui-même, où on peut introduire, en simulation tout au moins dans une première étape, ce qu'on suppose constituer une amélioration quant à l'efficacité assurée par cet algorithme.

b/ Selon le point de vue "temps".

En plus des informations ayant trait au point de vue "espace de mémoire", on peut recueillir des renseignements relatifs au temps simulé. Ceux-ci sont d'ailleurs analogues à ceux mentionnés lors du paragraphe a/ consacré aux outputs de la première méthode, et concernant les différents chemins qu'il est possible de suivre à l'intérieur de l'algorithme de remplacement.

Il est d'autre part naturel de profiter du fait que le modèle a le contrôle complet des segments pour dégager à propos de chacun de ceux-ci différents renseignements :

- le temps minimum, maximum, moyen de présence de segment en mémoire centrale et en mémoire auxiliaire.
- le temps total de présence du segment dans les deux mémoires mentionnées ci-dessus.

- la valeur des rapports $\frac{T_a}{T_t}$, $\frac{T_b}{T_t}$, avec :

$(T_a , T_b \leq T_t)$

T_t : temps global de présence du segment en mémoire centrale.

T_a : valeur du temps simulé pendant lequel le segment était de type A.

T_b : valeur du temps simulé pendant lequel le segment était de type B.

Ces résultats sont à mettre en liaison, lors de l'analyse, avec les propriétés et caractéristiques mêmes des segments (taille, possibilité de swapping, ...),de façon à pouvoir déterminer l'effet de celles-ci sur l'efficacité avec laquelle la structure adoptée pour ces segments contribue à l'acquisition de bonnes performances.

Quant aux différents outputs relatifs au temps simulé nécessaire pour le parcours des phases de l'algorithme, ils interviennent dans l'évaluation de l'efficacité assurée par celui-ci. Les conclusions que l'on peut en tirer doivent déboucher sur une décision de statu-quo ou de modification, laquelle sera nécessairement appliquée à la version réelle de l'algorithme ; en effet, étant donné que le modèle utilise uniquement comme paramètres externes les distributions des temps relatifs aux différents chemins, et il est hors de question qu'il puisse avoir sur elles une action directe qui soit justifiée et efficace.

B) Le cas des mémoires auxiliaires et des canaux

Etant donné que cette étude n'a pas donné lieu à l'élaboration d'une méthode de simulation dont le produit

pourrait dégager des critères de gestion optimale des mémoires secondaires, l'ensemble des outputs dont l'exposé va suivre est assez réduit .
Cela ne signifie nullement qu'une étude envisagée dans le sens mentionné ci-dessous n'a pas sa raison d'être ; en effet, elle entre dans le cadre de l'évaluation du Data Management plus particulièrement, et à ce titre doit faire partie intégrante d'un projet d'estimation des performances d'un O.S. qui se veut exhaustif.

Les résultats ci-dessous concernent l'occupation des différentes entités pendant le temps simulé total ; on peut avoir :

- un "trace" de la simulation donnant l'état de flags d'occupation des dispositifs.
- le pourcentage global d'occupation du CPU.
- le pourcentage global d'occupation de chaque canal (vers les disques, vers les bandes et autres mémoires).
- le rapport du temps simulé pour les opérations I/O sur le temps simulé total.
- selon chaque type de mémoire auxiliaire (disques, bandes, ...) le pourcentage global d'occupation simultanée
    - du CPU et du(des) canal(canaux) associé(s) aux disques.
    - du CPU et du(des) canal(canaux) associé(s) aux bandes.
    - du CPU et du(des) canal(canaux) associé(s) aux bandes et aux disques.
- le pourcentage global d'occupation de chaque "I/O device".
- le pourcentage global de recouvrement du CPU et de chaque "I/O device".
- le pourcentage global de recouvrement du CPU et de chaque type d'"I/O device".

Il peut être utile de prévoir également :

- le nombre minimum, moyen et maximum d'opérations simultanées sur disque.
- le nombre minimum, moyen et maximum d'opérations simultanées sur bande.

4.2.3. Outputs relatifs à la gestion des événements

Les valeurs statistiques relatives aux événements générés et présentant un certain intérêt sont les suivantes :

- pour chaque événement, le nombre total de ses occurrences.
- le temps moyen et total du déroulement de chaque phase logique (à chacune d'elles on a associé un événement déterminé).

pour chaque événement, la valeur du rapport

$$\left(\sum_{i=1}^{m} t_i\right) / T$$

avec :

- m : nombre d'occurrences d'un événement déterminé.
- t_i : pour l'occurrence n° i de l'événement, temps simulé de déroulement de la phase logique à laquelle est associé cet événement.
- T : temps simulé total.

On peut éventuellement adapter ce rapport pour chaque type de processus et pour chaque chaîne de traitement introduite dans l'input du modèle, si du moins les responsables souhaitent tirer des conclusions en s'appuyant sur de telles bases.

- - - - -

4.3. PROBLEME DE L'ACQUISITION DES RESULTATS

Le concepteur et réalisateur du modèle doit résoudre le problème de l'acquisition des différentes statistiques. Etant donné que la plupart des entités traitées possèdent des attributs dont la valeur est temporaire, il est nécessaire de saisir au moment opportun toute modification réalisée dont le résultat est susceptible d'influencer le comportement que l'on veut étudier. On peut distinguer deux types de simulateur, où les facilités offertes pour le calcul statistique sont quelque peu différentes :

a) un simulateur sans horloge : étant donné que les opérations effectuées ne se réfèrent pas à une valeur du temps simulé, il n'est pas possible de se servir de celui-ci pour élaborer les valeurs désirées. En conséquence, chacune des opérations doit être accompagnée, si du moins c'est utile, de sa contribution à la construction de l'output du modèle.

b) un simulateur avec horloge : dans un tel modèle, tout est relatif à une ou plusieurs valeurs du temps simulé. Pour les opérations qui n'agissent pas sur celui-ci, telles que la reconnaissance d'occupations de ressources par exemple, il est possible à certaines conditions, de concentrer une séquence d'instructions dont le but est d'élaborer de manière significative les statistiques désirées : il suffit pour cela d'établir une fréquence régulière d'exécution de cette séquence, mais qui soit suffisamment élevée que pour préserver l'aspect significatif des valeurs relevées. Par contre, en ce qui concerne les opérations génératrices d'intervalles ou de valeurs absolues de temps simulé, il est nécessaire de prélever au(x) moment(s) opportun(s) leur contribution aux résultats désirés.

C O N C L U S I O N S

CONCLUSIONS

"The key to performance evaluation as well as to systems design is an understanding of what systems are and how they work" (Calingaert [4]). Cette phrase situe l'optique dans laquelle l'évaluation des performances d'un O.S. est envisagée au long de cette étude. La complexité d'un système d'exploitation et la diversité des fonctions qu'il remplit nécessitent l'élaboration d'une synthèse de celui-ci de manière que les responsables puissent formuler en toute connaissance de cause les exigences auxquelles doit satisfaire un modèle d'évaluation. L'objectif poursuivi dans ces notes était d'envisager une approche par simulation du problème "Comment estimer l'efficacité du système existant ?". Le schéma figurant en page 24 apporte une réponse en deux points :

- le premier est relatif à une simulation du système exécutée à posteriori.
- le second suggère une simulation conséquente à la première, et qui peut être réalisée soit à priori, soit à posteriori.

La première phase de la procédure a pour but de dégager des normes de comportement relatives aux mécanismes étudiés plus spécialement ; ceux-ci, dans le cadre de l'évaluation globale d'un système, doivent constituer le résultat d'un choix qui rencontre le mieux possible le point de vue de l'utilisateur. Néanmoins, les critères que peut donner celui-ci ne sont pas forcément significatifs au niveau des entités que gère le système. C'est pourquoi il est souhaitable d'introduire dans le modèle un élément qui puisse établir le lien entre le domaine des chaînes de traitement et le domaine de la gestion des processus et des ressources ; la forme sous laquelle cette liaison est proposée dans le cadre de ce travail n'est autre qu'une schématisation fonctionnelle du workload, qui respect l'aspect structurel et quantitatif des applications réelles. Le déroulement de la simulation est effectué sur cette base et la génération des outputs doit avoir pour critère la sortie de résultats

qui permettent de procéder à une analyse et une interprétation qui soient fonction des différents degrés d'intérêt. Lorsqu'il est opportun de faire le bilan sur les performances du système, les responsables ont la possibilité d'initialiser la seconde phase de la procédure : en effet, les conclusions qu'ils tirent à ce moment ne sont rien d'autre qu'une raison d'introduire un ensemble de modifications qui constituent en fait tout ou partie d'une boucle de feedback (1) selon le fait que ces modifications s'appliquent soit au modèle uniquement, soit à l'O.S. lui-même et donc également au modèle. Dès lors que ces opérations sont réalisées, la seconde phase se poursuit par l'exécution des procédures d'évaluation du système, appelée à livrer de nouveaux résultats sur base desquels il sera possible de prendre de nouvelles décisions.

Laissons de côté maintenant le schéma global pour examiner quelque peu à partir des mécanismes mis en jeu, les possibilités offertes par une approche de ce type ainsi que les difficultés et inconvénients qu'elle laisse apparaître.

+) L'utilisation de la simulation à des fins d'évaluation de performances est incontestablement, en raison de la souplesse de la méthode, un choix judicieux. Un exemple probant parmi d'autres de la manifestation de cet avantage réside dans le fait que l'emploi d'un modèle de ce type est possible aussi bien à priori qu'à posteriori, ce qui n'est pas réalisable avec une technique de mesures. Par contre, la définition du niveau de simulation peut poser quelques problèmes, et la période de conception - implémentation - exploitation du modèle peut se révéler si importante que les résultats obtenus après son écoulement présentent une opportunité négligeable sinon nulle.

+) La façon dont on a envisagé la représentation de la configuration et de la charge simulée de travail aussi bien que leur utilisation et essentiellement déterministe,

---

(1) Noetzel [6] considère que la boucle de feedback (qu'il appelle "Méta-Système") comprend les fonctions de mesure, d'évaluation et de modification.

dans une très grande partie tout au moins. Cette optique est liée à une décision visant à traduire la réalité de manière telle qu'il soit possible de préserver la structure et les quantités qui la caractérisent. Une conséquence heureuse et d'ailleurs immédiate de cette option est la possession par le modèle du contrôle complet de toutes les opérations générées ou exécutées ainsi que du contexte qui entoure celles-ci. Il est clair que l'obtention d'une bonne schématisation destinée à constituer l'input de l'évaluation n'est pas une tâche très facile, et qu'à ce propos un problème de validation est soulevé ; d'autre part, le coût engendré par la maintenance du contrôle complet assuré par le modèle sur l'environnement simulé risque d'être assez important, compte tenu du fait que ce contrôle ne sert à rien si ses résultats ne sont pas communiqués au responsable de l'étude. Néanmoins, l'information qu'il est possible de recueillir grâce à ce dispositif présente un intérêt majeur : celle de pouvoir donner la valeur d'une caractéristique quelconque de n'importe quelle entité à n'importe quel moment du temps simulé. De ce fait, un "trace" détaillé de la simulation, un état complet de la mémoire centrale, l'occupation de telle ou telle ressource, l'état d'un processus donné, la valeur de l'horloge, la position d'un tel segment, ... peuvent être connus à tout instant par l'analyste de système ; c'est à lui de préciser la nature des informations dont il estime l'utilité comme étant suffisamment importante.

+) Le principe de la simulation par événement permet au concepteur et au réalisateur de disposer d'une grande liberté d'action quant à l'organisation elle-même du déroulement de la simulation. Un événement est associable à tous les phénomènes qui existent et à toutes les opérations que l'on désire introduire dans le modèle. Le choix est laissé entièrement au réalisateur pour décider de la contribution ou non d'un tel événement au progrès des processus simulés ou à la constitution de valeurs statistiques ; ce qui met

en évidence le caractère multivalent d'une telle structure, pour autant du moins qu'on sache utiliser celle-ci avec les précautions nécessaires. Cette propriété possède d'ailleurs un domaine d'application qui ne se limite pas à l'exemple ci-dessus, mais s'étend au-delà de la solution à un problème de pure organisation. En effet, la notion de simulation par événement, liée à un aspect de modularité, permet d'affecter au modèle deux dimensions, horizontale et verticale, qui répondent le plus exactement possible aux souhaits de l'analyste de système. La première d'entre elles traduit l'éventail des fonctions d'un O.S. dont on a décidé l'introduction dans le modèle tandis que la seconde représente le niveau de détail auquel on a jugé utile de descendre dans l'estimation des performances. Selon l'importance de telle fonction, selon le volume de renseignements que le responsable attend de la simulation de telle autre, l'existence (ou l'inexistence) et le rôle des événements seront conçus de manière à satisfaire dans les meilleures conditions les buts et exigences de l'évaluation. Il n'est pas inutile d'insister sur l'importance capitale de cet aspect de multivalence que possède une structure de simulation par événement, car il s'agit probablement là de la propriété la plus puissante qui caractérise l'approche développée au cours de cette étude.

Ceci termine l'examen des avantages et inconvénients que l'on peut déceler après avoir considéré de façon synthétique la méthode proposée ; il est clair qu'il ne s'agit pas là d'une critique détaillée ayant pour but de faire apparaître le pour et le contre de chaque mécanisme particulier utilisé, une telle option ne constituant pas le but de ce travail.

L'intérêt s'est porté sur les bases de la méthode, et plus spécialement sur les possibilités et les limites inhérentes à celles-ci. La raison en est simple, et tient au fait que l'objectif de cette étude était de tenter d'élaborer certains fondements de l'évaluation des performances d'un operating system reposant sur le concept

de processus.

Quant au problème lui-même, il ne s'en trouve pas résolu pour autant, loin s'en faut.
Sa dimension et sa complexité sont telles que la recherche et l'implémentation d'une solution satisfaisante nécessitent la mise en oeuvre, dans un contexte approprié, de moyens humains, financiers et autres dont l'importance se situe assez loin d'un niveau relativement peu élevé.

*

* *

B I B L I O G R A P H I E

B I B L I O G R A P H I E

[1] : L.R. HUESMANN and R.P. GOLDBERG : "Evaluating computer systems through simulation". The Computer Journal Vol. 10 (May 1967 - February 1968).

[2] : H.C. LUCAS : "Performance Evaluation and Monitoring". Computing Surveys - Vol. 3 - n° 3 - September 1971.

[3] : R.M. GRAHAM : "Performance Prediction". Technical University of Munich - Germany Advanced Course on Software Engineering - February 21, March 4, 1972.

[4] : P. CALINGAERT : "System Performance Evaluation : Survey and Appraisal". Communication of the ACM - Vol. 10 - Number 1 - January 1967.

[5] : C.C. GOTLIEB : "Performance Measurements". Technical University of Munich - Germany - 1972.

[6] : A.S. NOETZEL : "The design of a Meta-System". Spring Joint Computer Conference 1971. AFIPS Conference Proceedings - Vol. 38 - Pages 415 -424

[7] : S.L. REHMANN and S.G. GANGWERE, Jr. : "A simulation study of resource management in a T.S. system". Fall Joint Computer Conference 1968 AFIPS Conference Proceedings - vol. 33 - Part II - Pages 1399 - 1410.

[8] : J.E. BELL, B.W. BOEHM, R.A. WATSON : "Framework and initial phases for computer performance improvement". Fall Joint Computer Conference 1972 AFIPS Conference Proceedings - vol. 41 - Part II -

[9] : M.H. Mac DOUGALL : "Computer System Simulation : An Introduction". Computing Surveys - Vol. 2 - Number 3 - September 1970

[10] : S.R. KIMBLETON : "Performance Evaluation : A Structured Approach". Spring Joint Computer Conference - 1972 AFIPS Conference Proceedings - Vol. 40 - Pages 411 - 416

[11] : F.F. MARTIN : "Computer Modeling and Simulation". John Wiley and Sons, Inc. - New York - London - Sydney - 1968.

[12] : T. WILLIAMS : "Computer Systems Measurement and Evaluation". The Computer Bulletin - February 1972.

[13] : R.A. MERIKALLIO - F.C. HOLLAND : "Simulation Design of Multiprocessing System". Fall Joint Computer Conference - 1968 AFIPS Conference Proceedings - Vol. 33 - Part 2.

[14] : D.C. WOOD and E.H. FORMAN : "Throughput Measurement using a Synthetic job stream". Fall Joint Computer Conference - 1971 AFIPS Conference Proceedings - Vol. 39

[15] : COMRESS Incorporated : "SCERT Users Manual", Washington, D.C., March 1965.

[16] : "OPERATING SYSTEMS SURVEY" by the Comtre Corporation. Edited by Anthony P. SAYERS. Auerbach Computer Sciences Series.

[17] : G.P. BLUNDEN and M.S. KRASNOW : "The Process concept as a Basis for Simulating Modeling" - IBM Corporation - IBM - ASD - TR - 17 - 181, Yorktown Heights, N.Y. - 1965

[18] : S.S. PATIL : "Coordination of asynchronous events". Project MAC Technical Report TR - 72 - MIT - Cambridge - Massachussetts - June 1970